Jean-François BERNARD
Jean-Yves MARIN
Édouard BOUCHER
Olivier LONZI



IUT Fontainebleau 77300
Académie de Créteil

IRIS-FIGHT

RAPPORT DE PROJET

Tuteuré par M. Luc HERNANDEZ

Licence Professionnelle – Session 2008
Système **I**nformatique et **L**ogiciel
Conception et **M**aintenance de **L**ogiciel **L**ibre



I. Sommaire

II. Présentation des personnes en charges du projet

Tuteur : Luc HERNANDEZ

M. HERNANDEZ nous à soutenus dans notre démarche pour la conception et réalisation de notre projet. Nous le remercions pour ses conseils avisés, notamment pour l'architecture du rapport.

Étudiant n°1 : Édouard BOUCHER

Édouard s'occupe de la partie fenêtrage du logiciel, pour les clients.

Étudiant n°2 : Olivier LONZI

Olivier s'occupe de la liaison client/serveur, et du serveur.

Étudiant n°3 : Jean-Yves MARIN

Jean-Yves s'occupe de la partie zone de combat, programmé en SDL.

Étudiant n°4 : Jean-François BERNARD

Jean-François s'occupe de la partie WEB. Il est arrivé après la définition de notre cahier des charges.

Pour identifier les personnes ayant travailler sur telle ou telle partie du projet, nous avons choisi d'utiliser un jeu de couleur, un par personne (voir ci-dessus), que nous reportons à chaque fois que des travaux personnels ont été réalisés.

III. Introduction

IrisFight est un logiciel fonctionnant sous Windows et Linux et est développé en C++ avec les librairies Qt et SDL. Il est composé d'une partie serveur et d'une partie client, permettant respectivement d'héberger un jeu et de s'y connecter. Le logiciel dispose également d'une extension Internet développé en PHP permettant de créer ou de se connecter à son compte personnel, de modifier ses caractéristiques ou encore de télécharger les dernières versions du jeu.

Le principe de ce logiciel est un jeu de combat en tour par tour durant lesquels l'utilisateur choisit ses attaque en combinant différentes orbes magiques, chaque combinaison ayant des effets distincts. A l'issue de chaque combat, les personnages obtiennent de l'expérience leurs permettant d'augmenter de niveau et ainsi de se spécialiser dans des domaines spécifiques.

Pour ce faire, l'utilisateur peut dépenser ses points durement acquis grâce à la puissance de ses sorts, lors d'un passage de niveau dans les six catégories offensives disponibles que sont la terre, l'air, l'eau, le feu, la lumière et les ténèbres. Il peut aussi les dépenser dans une catégorie correspondant à son niveau de vie.

Les différentes catégories sont symbolisées par des orbes (voir image 1), distribués au hasard au joueur durant la partie: ils lui permettent d'invoquer des sorts plus ou moins puissants.

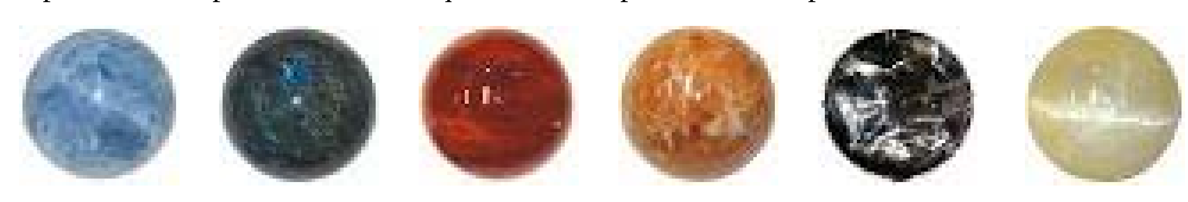


image 1 : Représentation des orbes {Air, Eau, Feu, Terre, Ténèbres, Lumière}

Ainsi, il existe trois types de sorts, les sorts défensifs, les sorts offensifs et les sorts de défense offensifs (voir annexe «Principe de base de l'alchimie des orbes» p 37).

- Les sorts défensifs nécessitent deux orbes de même catégorie.
- Les orbes offensifs nécessitent 3 orbes minimum, dont deux de la même catégorie. L'orbe unique définit l'élément du sort, ainsi que les effets secondaires qui s'ensuivront. Les deux orbes identiques définissent le domaine, pouvant en augmenter considérablement les effets (voir annexe «Domaines et effets secondaires» p 38).
- L'attaque la plus puissante d'un personnage est «le pentacle»: il correspond à une suite de cinq orbes de même catégorie.

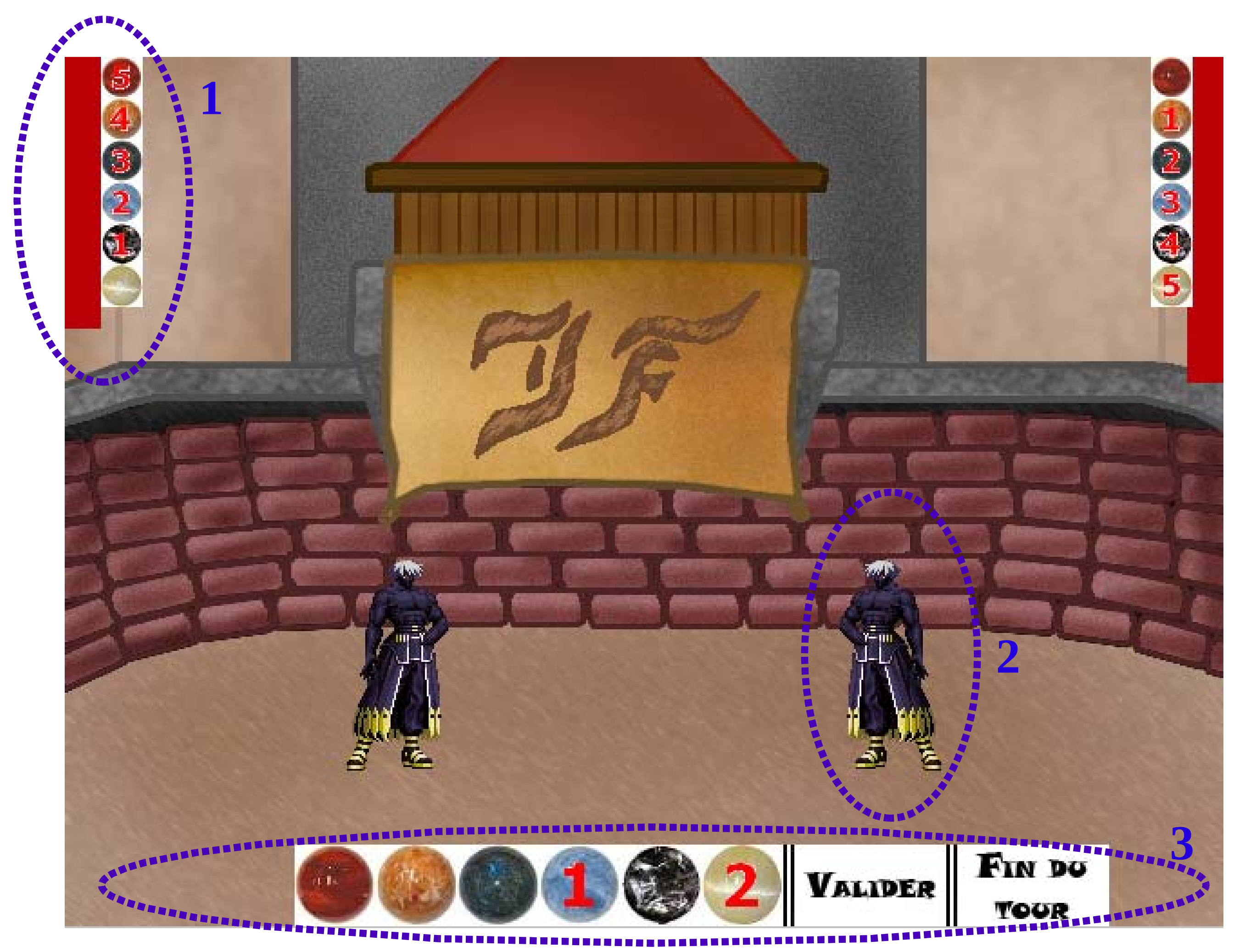


image 2 : Arène de combat : Choix des actions

1. Liste des orbes disponibles du 1er joueur, ainsi que sa vie. (Les mêmes informations sont disponibles pour le 2nd joueur dans le coin opposé de l'écran).
2. Représentation du 2nd joueur, ici la même image que pour le premier joueur.
3. Barre d'action: permet au joueur de sélectionner plusieurs combinaisons d'orbes en validant chacune d'elles puis une fois toutes les actions validées, de terminer le tour.



image 3 : Arène de combat présentant un sort de feu

Un duel se termine par la perte totale des points de vie d'un personnage ou l'abandon d'un des joueurs. A l'issue de chaque duel, le vainqueur gagne trois points d'expérience, tandis-que le vaincu n'en gagne qu'un (voir annexe «Expérience» p 40).

Une fois un certain seuil d'expérience atteint (voir image 4 – Rappel de l'annexe), l'utilisateur peut passer au niveau suivant, et répartir les points d'orbes acquis dans les catégories de son choix.

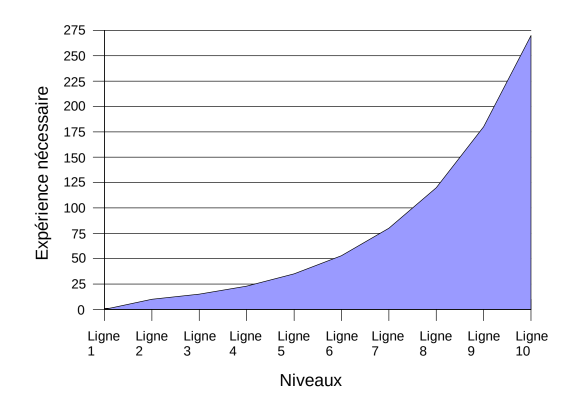


image 4 : courbe de passage de niveau

En dehors des phases de combat, les utilisateurs ont accès à un chat regroupant tous les joueurs connectés au serveur, identifiés par leurs pseudonymes respectifs. Ils peuvent ainsi discuter avec entre-eux mais aussi se défier en duel. Bien entendu, le joueur défié a la possibilité d'accepter ou non le duel.

IV. Dossier de Spécification

1. Rappel du cahier des charges

Ce projet consiste à réaliser un jeu en deux dimensions dans lequel deux joueurs s'affrontent chacun à leur tour.

Les joueurs se connectent à un serveur de combat, dans lequel ils créent ou ont déjà créé un personnage. Ces personnages ont différentes caractéristiques (air, terre, feu, eau, ténèbres, lumière, vie) et peuvent moduler leur efficacité en fonction des points de compétence acquis.

Les joueurs apparaissent dans la liste des joueurs connectés et peuvent proposer un combat à tous les autres joueurs de la liste ou choisir de discuter avec eux.

Une fois qu'un joueur en a défié un autre et que celui-ci accepte le duel, un combat commence entre leurs deux personnages.

L'issue du duel se détermine par la perte de tout les points de vie d'un des deux personnages ou par l'abandon d'un des joueurs.

Une fois le duel terminé, des points d'expérience sont gagnés, permettant ainsi aux joueurs de faire évoluer les caractéristiques de leurs personnages.

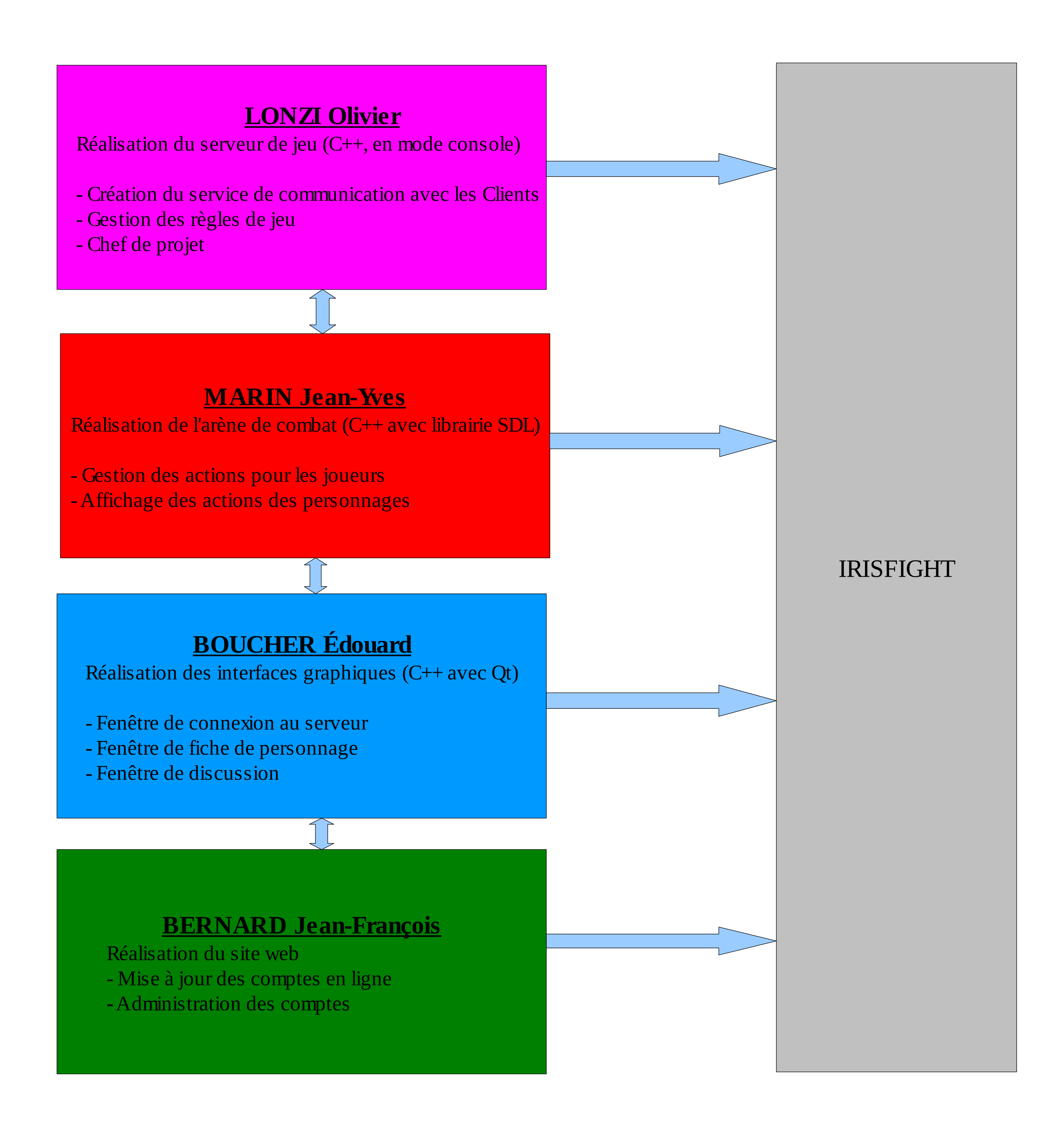
2. Solutions techniques choisies

Pour réaliser ce projet, nous avons fait le choix d'utiliser les matériaux suivants :

- SDL pour le moteur de jeu (voir chapitre SDL en annexe p 44)
- Qt pour les menus (voir chapitre QT en annexe p 46)
- PHP/MySQL pour le site internet (voir chapitres PHP en annexe p 51 et MySql en annexe p 50)
- C++/MySQL pour le serveur (voir chapitres C++ en annexe p 49 et MySql en annexe p 50)

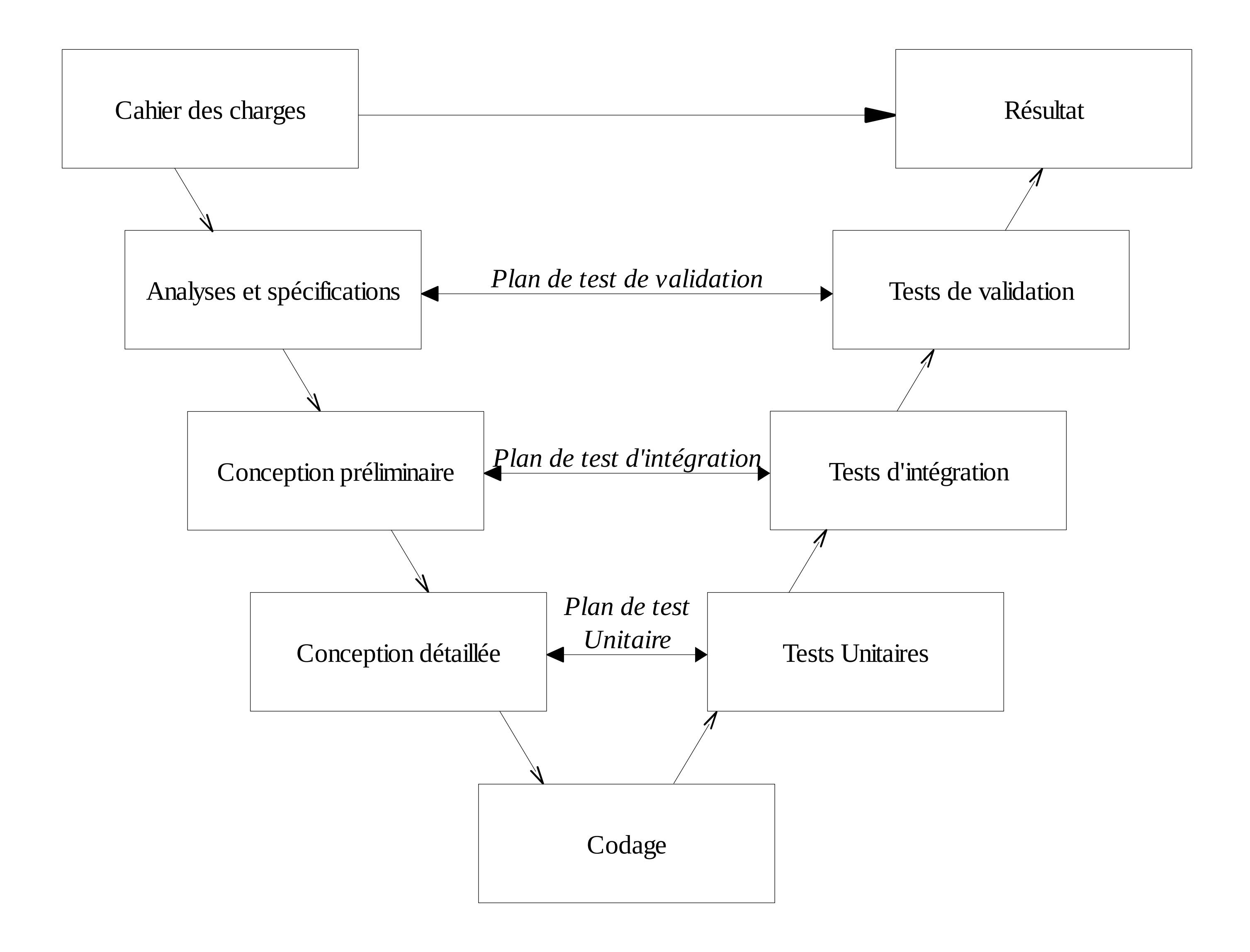
3. Répartition des tâches

Afin de travailler sur un jeu complet, il était important de dissocier les tâches, de manière à pouvoir ainsi regrouper plus facilement l'ensemble de nos travaux.



4. Méthodologie de travail en groupe

Afin de réaliser ce projet en groupe, nous avons opté pour l'utilisation du cycle en Vé, que nous avons appris tout les trois en BTS IRIS. Jean-François s'étant rattaché au groupe par la suite, a accepté cette méthodologie simple, que nous lui avons expliquée.Nous l'aidons dans la réalisation de ses parties.



Le Diagramme du Cycle en Vé se lit de haut en bas et de la gauche vers la droite. Nous avons dans un premier temps réalisé le cahier des charges (voir annexe p 62), puis réalisé l'analyse et la spécification des besoins, déterminé la conception préliminaire et la conception détaillée, ainsi que certains plans de tests associés. Ces différentes parties représentent les grands axes de ce rapport.

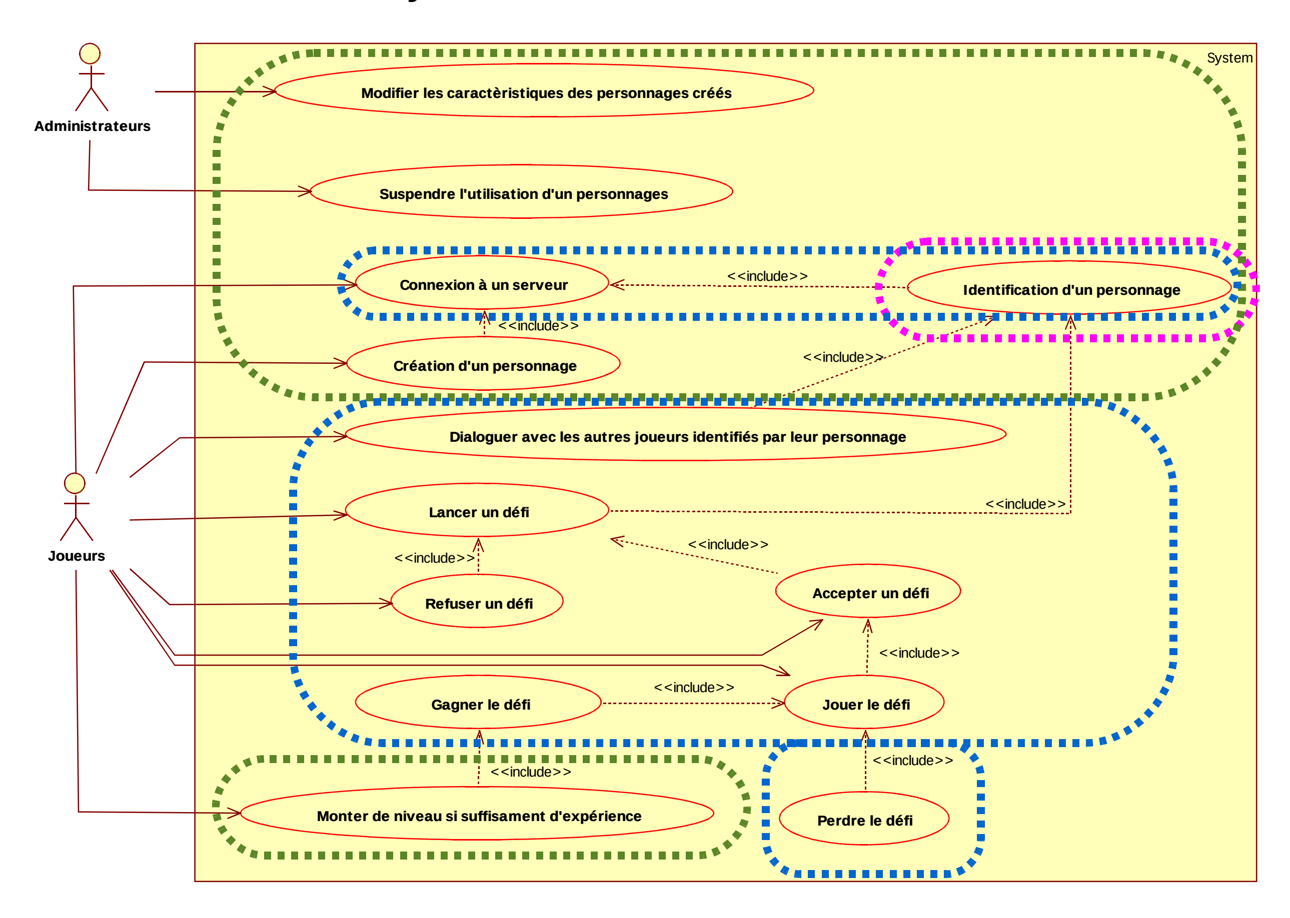
V. Analyses & Spécifications

1. Marche à suivre pour un joueur

Explication textuelle reprenant le cahier des charges, permettant d'aboutir sur les diagrammes UML, en reprenant les points nécessaires à un client pour jouer à notre jeu.

1. Lancement du programme client.
2. Saisie de l'identifiant et du mot de passe.
3. Envoie des informations d'identification au serveur et acceptation ou refus de celui-ci.
4. Refus : Re-saisie des informations (étape 2).
5. Acceptation : Suite de la procédure.
6. Ouverture de la page de discussion en ligne avec les autres joueurs.
7. Le joueur peut saisir des messages et les envoyer à toutes les personnes connectées sur le serveur.
8. Il peut choisir de défier d'autres joueurs ou d'accepter/refuser un défi.
9. Refus : Le joueur peut continuer à saisir des messages (étape 5)
10. Acceptation : La phase de combat débute pour les deux joueurs.
11. Les joueurs choisissent les actions à effectuer, puis les transmettent au serveur.
12. Le serveur trie les actions et teste les points de vie pour définir la fin de partie.
13. Tant que les joueurs ont de la vie et qu'aucun des deux n'abandonne, le jeu continue (étape 7)
14. Lors de la fin de la partie, des points d'expériences sont attribués aux joueurs.
15. Les joueurs retournent sur la fenêtre de discussion.
16. Si le joueur a assez d'expérience, une notification lui annonce qu'il peut monter de niveau.
17. Le joueur peut donc ouvrir la fenêtre de gestion des niveaux et attribuer ses points dans ses caractéristiques.

2. Cas d'utilisation du système client



Le joueur dispose d'un panel d'action assez important durant le jeu, il peut ainsi créer un personnage via le site web, se connecter à un serveur, dialoguer avec les autres clients, lancer ou refuser un défi, jouer le défi puis gagner de l'expérience (et par extension, des niveaux), pour améliorer son personnage.

L'administrateur, quant-à lui, dispose de tout les droits sur le système, il est donc habilité à faire la totalité des actions des joueurs. Il dispose également de deux choix d'actions supplémentaires : Modifier les caractéristiques de n'importe quel personnage et suspendre l'utilisation d'un personnage (en cas de tricherie par exemple).

3. Diagramme de déploiement client/serveur

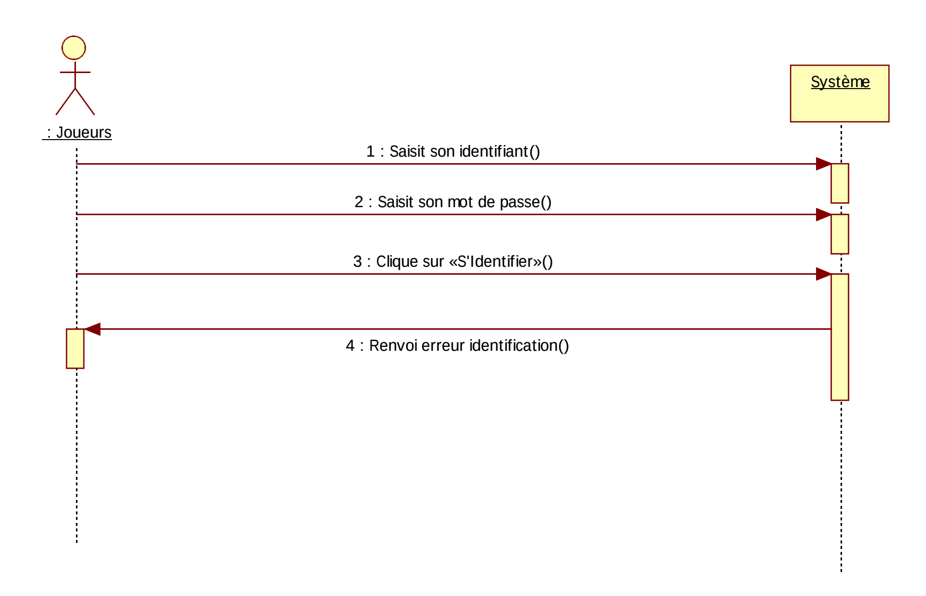


Le serveur peut supporter une infinité de clients connectés simultanément (la limite maximum est choisie dans le fichier de configuration du serveur et ne peut dépasser 4294967296 – entier long non signé).

4. *Diagrammes des scénarios*

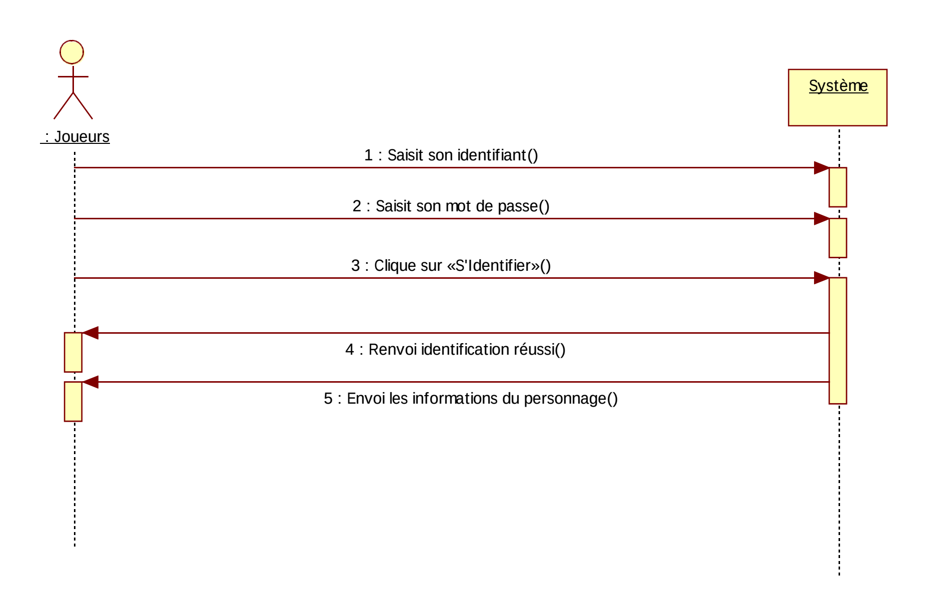


1. Échec d'identification



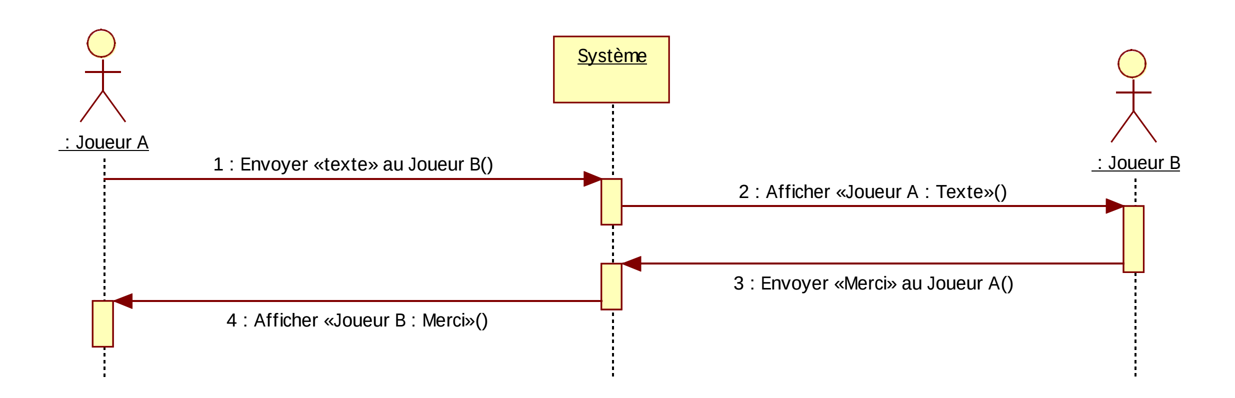
Le joueur entre ses informations de login et mot de passe: s'ils sont incorrects, le serveur renvoi une erreur.

2. Réussite d'identification



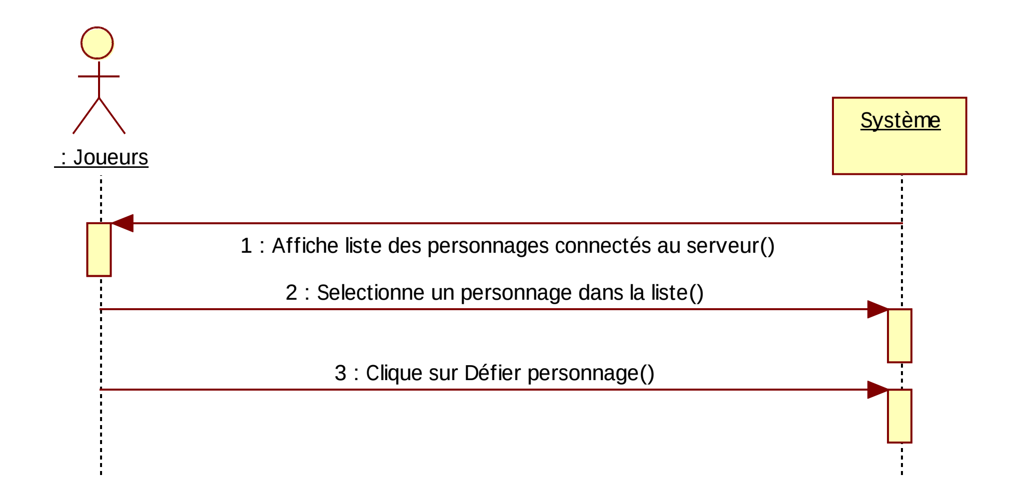
Le joueur entre ses informations de login et mot de passe: s'ils sont corrects, le serveur accepte la connexion et lance la fenêtre de chat.

3. Dialogue entre deux joueurs



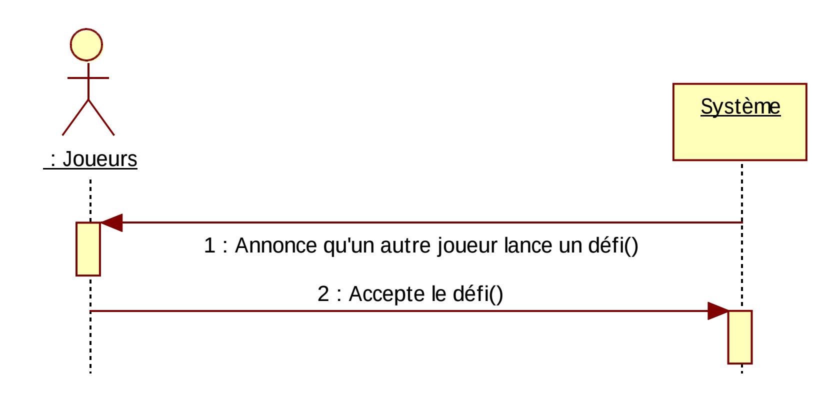
Le joueur A envoi un message au système, qui relègue l'information au joueur B, et ainsi de suite.

4. Un joueur lance un défi à un autre joueur



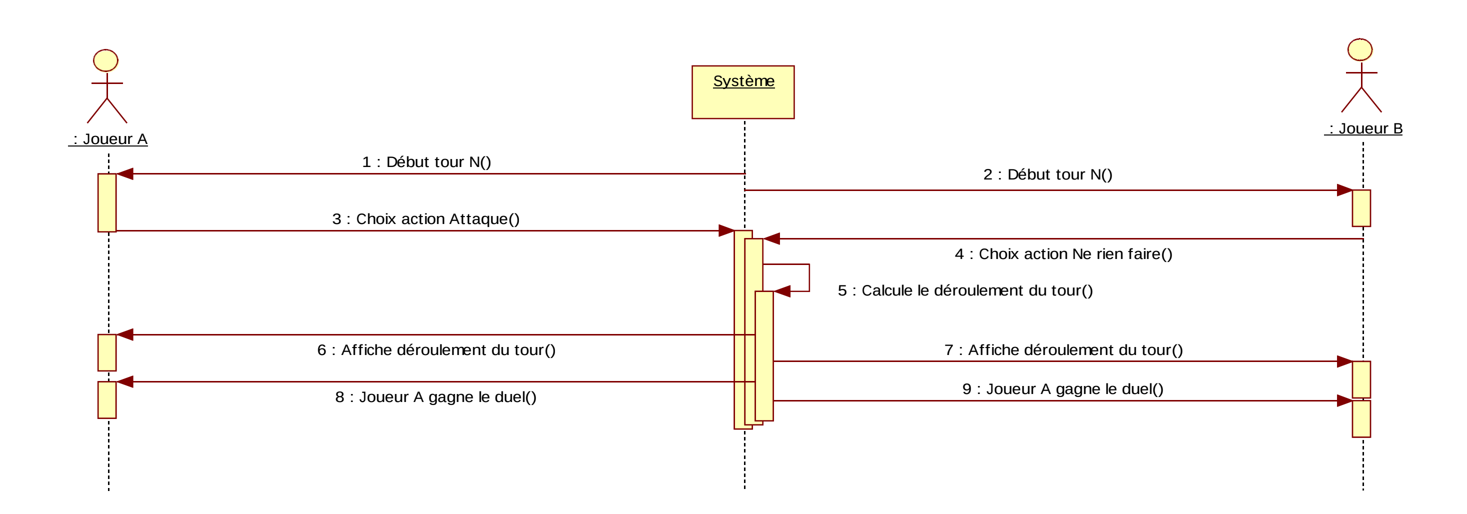
Le système envoie la liste des clients connectés au serveur. Le client peut donc sélectionner un autre client et cliquer sur le bouton « défier » pour lancer la demande de défi au joueur adverse.

5. Un joueur accepte un défi



Le serveur annonce au client qu'un adversaire vient de le défier. Le client accepte en cliquant sur le bouton « accepter ».

6. Gagner un défi



1. Le serveur annonce aux Joueurs A et B le début du tour N. Le joueur A décide alors de l'attaque à mener, tandis que le joueur B décide de l'action à faire, ou à ne pas faire. Le serveur récupère et traite les informations puis affiche aux joueurs A et B le déroulement du tour. A la fin de la partie, le serveur annonce aux deux joueurs que le joueur A a remporté le duel.

5. La base de données MySQL

1. Pourquoi une base de données

Afin de stocker les comptes des joueurs et de pouvoir sauvegarder l'évolution des personnages, nous avons décidé d'utiliser une base de donnée (voir le chapitre «Pourquoi MySQL» p 50).

2. Qui utilise la base ?

Deux parties du projet utilise la base de données, le site Internet et le serveur de jeu, respectivement réalisés par Jean-François BERNARD et Olivier LONZI.

3. La table de la base de donnée

persos
<<primaire>>ID_perso: int(10) actif: int(1) = 1 admin: int(1) = 0 joueur_prenom: varchar(50) joueur_nom: varchar(50) <<unique>>nom: varchar(50) pass: varchar(50) courriel: varchar(100) niveau: int(2) = 1 experience: int(10) point_vie: int(2) = 1 point_orbe_1: int(2) = 1 point_orbe_2: int(2) = 1 point_orbe_3: int(2) = 1 point_orbe_4: int(2) = 1 point_orbe_5: int(2) = 1 point_orbe_6: int(2) = 1

Cette table stocke toutes les informations relatives à un personnage, son niveau, son expérience ainsi que des informations personnelles sur le joueur rattaché au compte dont voici les explications :

- **ID_perso :** Cette valeur correspond à la clé primaire du personnage, il ne peut y avoir qu'une valeur identique.
- **actif :** Cette valeur correspond à l'état d'activation du compte (1 si actif, 0 sinon)
- **admin :** Cette valeur correspond au statut du compte (1 si administrateur, 0 sinon)
- **joueur_prenom, joueur_nom et courriel :** Ces valeurs contiennent toutes les données personnelles du joueur
- **nom et pass :** Ces valeurs correspondent à l'identifiant-nom du personnage ainsi qu'à son mot de passe (le nom étant une valeur unique dans la table)
- **niveau et expérience :** Ces valeurs correspondent au niveau et à l'expérience acquis par le personnage
- **point_vie et point_orbe_1 à point_orbe_6 :** Ces valeurs correspondent aux points d'orbe dépensés par le joueur dans les différentes compétences du personnage.

VI. Plan de Tests de Validation du système

Le Plan de Tests de Validation permet de définir des «actions à vérifier» en proposant différents tests possibles (incluant des valeurs précises si possible) et bien sûr en indiquant le résultat attendu pour chaque test. Ce tableau nous permettra, dans une étape finale, de vérifier points par points la corrélation entre ce que nous voulions et ce que nous avons obtenu.

Actions à vérifier	Tests effectués	Résultats attendus
Connexion à un serveur	Ip : 127.0.0.1	Afficher connexion réussie
	Ip : 127.0.0.1 – port 8000	Afficher connexion échec
	Ip : localhost (trouver l'ip suivant le dns)	Afficher connexion réussie
	Ip : test_dns_distante avec serveur distant configuré	Afficher connexion réussie
Création d'un compte	Saisir toutes ses infos dans le formulaire «créer un compte», sur le site puis valider.	Ajout d'une nouvelle ligne dans la table. Se connecter en mode administrateur pour le vérifier.
	Ne pas saisir toutes les informations.	Ne rien ajouter, indiquer «veuillez saisir tous les champs obligatoires»
Suspension d'un compte	Se connecter en admin, puis décocher la case «compte actif»	Modifier le compte et décocher la case, se connecter en mode admin pour vérifier.
	Saisir l'identifiant et mot de passe d'un compte préalablement désactivé.	Le serveur doit refuser la connexion.
Passage de niveau (10 points à répartir)	Le joueur répartit tous ses points	Modifie la table, avec les nouvelles valeurs.
	Le joueur ne répartit pas tous ses points	Ne modifie rien.
Connexion à un compte	Login : faux – mdp : faux	Afficher «Login ou mot de passe incorrect»
	Login : juste – mdp : faux	
	Login : faux – mdp : juste	
	Login : juste – mdp : juste	Ouvrir la fenêtre de tchat et afficher la liste des pseudos des joueurs connectés.
Discussion entre les joueurs dans la fenêtre de tchat	Saisir le message «Bonjour», dans la zone prévue, puis valider.	Tous les joueurs connectés reçoivent le message «Bonjour» de la part du joueur émetteur.

Actions à vérifier	Tests effectués	Résultats attendus
J1 défi J2	J2 déjà en combat	Ne rien faire
	J2 n'est pas en combat	Indiquer à J2 le défi de J1
J2 accepte le défi de J1	J1 n'a pas proposé de défi ou l'a annulé	Ne rien faire
	J1 a proposé un défi mais est en combat maintenant	L'indiquer à J2: défi annulé
	J1 à proposé un défi et n'est pas en combat.	Lancer la fenêtre de combat pour les deux joueurs.
J2 refuse le défi de J1	Quelque soit l'état de J1 (en combat ou non)	Indiquer à J1: défi refusé par J2.
Sort dans un combat	Choisir 3 orbes de feu	Lancer attaque de feu
	Choisir 2 orbes d'eau	Lancer sort défensif d'eau
	Choisir 3 orbes ténèbres puis 2 orbes terre.	Lancer sort défensif terre puis sort attaque ténèbres.
Fin de combat	Mort d'un joueur (amener la vie d'un joueur à 0, ou moins)	Indiquer fin de partie au deux joueurs et attribuer des points d'expérience.
	Abandon d'un joueur (Déconnexion volontaire ou non)	Indiquer fin de partie au joueur restant, en le déclarant vainqueur.

Les tests de validation sont effectués de façon visuelle lors de l'exécution du programme serveur et du programme client.

VII. Conception préliminaire

La phase de conception préliminaire consiste à définir les interactions entre les différentes parties du système, à travers les diagrammes de classes et de composants, en ajoutant si nécessaire des informations textuelles.

1. Définition des axes entre les étudiants

1. Édouard / Olivier

Afin de réaliser la liaison entre la partie client (*Édouard*) et le serveur (*Olivier*), nous avons définit les besoins suivants :

- Le client doit se connecter au serveur.
- La gestion des trames devra être uniformisée par un seul étudiant.

De ce fait nous avons mis en place un système de méthode logique à appeler par le client. En d'autres termes, *Olivier* fournit une classe à *Édouard* dans laquelle il existe des méthodes simples comme : EnvoyerMessage() ou DemanderListeClients().

La liste exhaustive des fonctions est disponible en annexe p 36.

Avec ce système nous avons défini un dialogue simple entre nos deux parties, évitant ainsi des intégrations complexes et fastidieuses (par exemple, si chacun avait réalisé son code mettant en oeuvre son socket, de plus, il n'est pas nécessaire que toutes les fonctionnalités soient implémentées pour réaliser les premiers tests. Pas besoin d'envoyer un message pour demander la liste des clients.

2. Jean-Yves / Olivier

VIII. Afin de lier la partie de *Jean-Yves* (Interface graphique) avec la partie d'*Olivier* (Serveur), il faut se référer à l'objet de communication et aux trames vues dans l'axe précédent (*Édouard* / *Olivier*). De ce fait, le système étant le même étant transparent, pour le serveur, il faut se référer à l'axe suivant, pour voir comment *Jean-Yves* peut utiliser les mêmes fonctions que *Édouard* pour dialoguer avec le serveur.

1. Édouard / Jean-Yves

Afin de faire la jonction des deux parties, *Édouard* envoi à *Jean-Yves* une fonction lui permettant de dialoguer avec le serveur. *Jean-Yves* transmet ensuite à *Édouard* sa réponse, que ce dernier se charge de faire suivre au serveur.

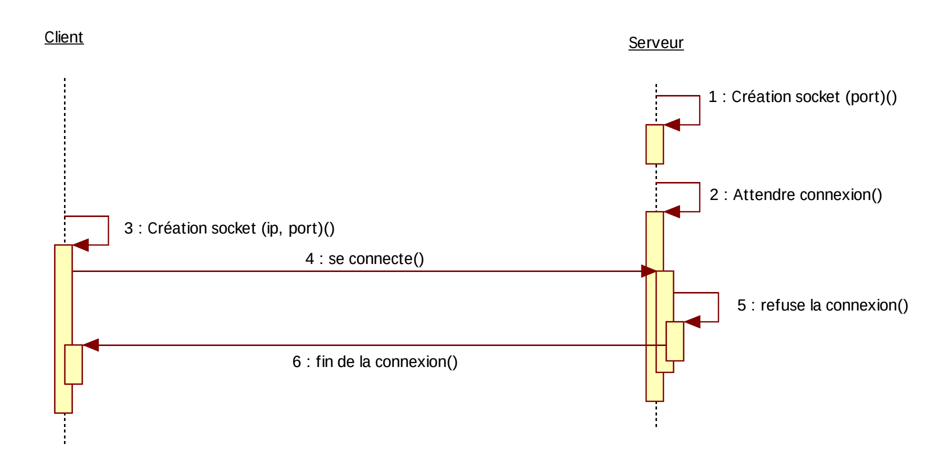
2. Jean-François / Olivier

La jonctions entre ces deux parties se fait intuitivement par rapport à la BdD: les deux parties peuvent modifier et/ou sélectionner les données des tables, pour leurs traitements respectifs.

2. Méthode du dialogue réseau {Olivier LONZI}

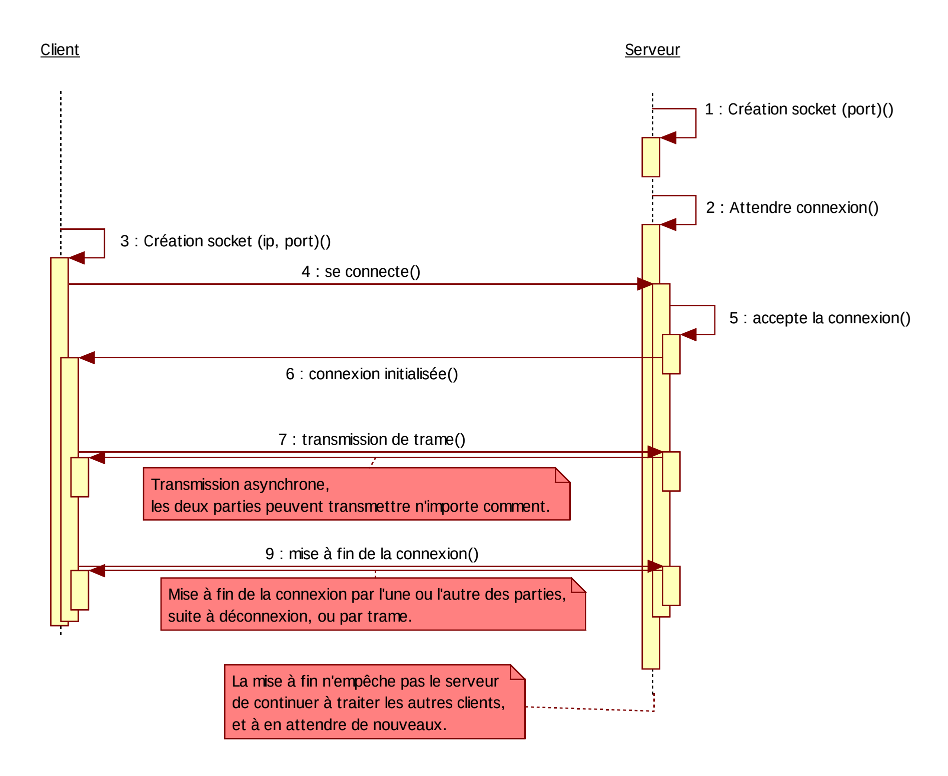
Pour dialoguer entre la partie serveur et la partie client, nous avons choisi les protocoles TCP et IP souvent appelés le protocole TCP/IP: ces protocoles sont utilisés via les méthodes de gestion des sockets en C.

1. Schéma de connexion d'un client (Refus)



Le serveur peut refuser une connexion, par exemple si le nombre de clients max est atteint.

2. Schéma de connexion d'un client (Acceptation)



Le client peut dialoguer de façon asynchrone avec un client et il peut à tout moment accepter ou refuser d'autres clients.

3. Le fonctionnement des trames {Olivier LONZI}

1. Introduction

La transmission des trames peut être représentée par une chaîne de caractères. Afin de pouvoir décoder n'importe quelles trames d'un coté comme de l'autre, et donc de savoir comment les encoder, j'ai utilisé un système de sérialisation de trames.

2. Le besoin

- Chaînes de caractères à dimensions variables à sérialiser.
  - Nécessite de gérer des trames de taille variable.
- Données variables, définissant un grand nombre d'informations, sans rapport.
  - Nécessite de différencier les trames d'un seul coup.

3. Exemple de trames

Afin de répondre aux besoins, j'ai mis en place un système de trame sous cette forme :

octet	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*
Ex 1	1	0	\n	**i**	**M**	**i**	**r**	**a**	**n**	**d**	**a**	**\n**	**a**	*	*	*	*	*	*	*	*	*	*
Ex 2	1	2	\n	**n**	**\0**	**\0**	**\0**	**\0**	**M**	**i**	**r**	**a**	**n**	**d**	**a**	*	*	*	*	*	*	*	*
Ex 3	7	\n	**i**	**T**	**e**	**s**	**t**	**\n**	**t**	*	*	*	*	*	*	*	*	*	*	*	*	*	*

- L'exemple 1 annonce qu'un client, précédemment accepté par le serveur, souhaite s'identifier et fourni son pseudonyme (Miranda) et son mot de passe (a).
- L'exemple 2 annonce que le joueur Miranda (numéro identifiant 0), vient de se connecter.
- L'exemple 3 annonce que le client Test tente de se connecter, mot de passe (t).
- **Attention :** Les étoiles blanches peuvent être n'importe quoi, et ne sont pas envoyées par l'expéditeur, ou alors c'est une seconde trame qui devra être analysée à part.

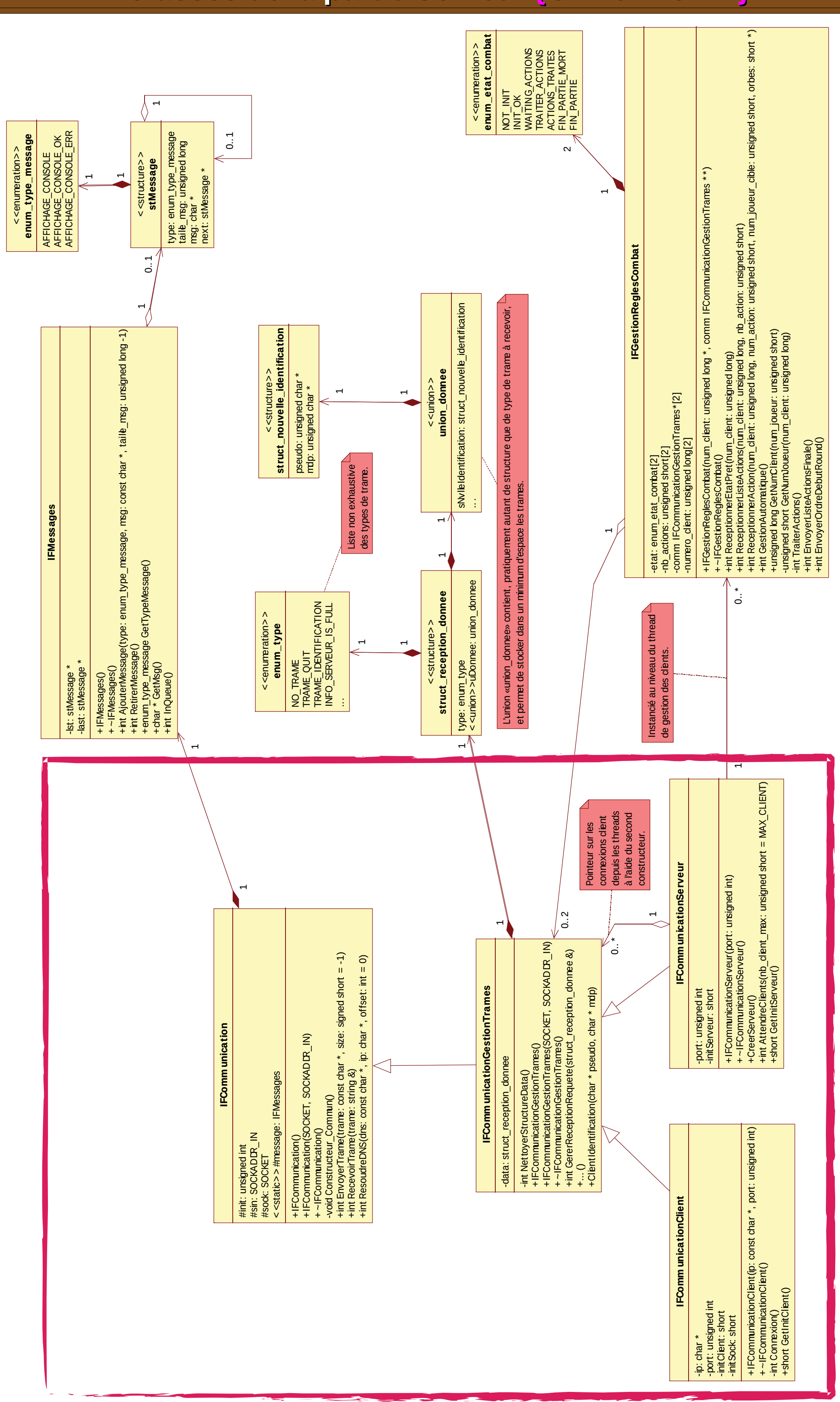
4. Comment décoder la trame

Le récepteur (client ou serveur) va dans un premier temps lire le buffer du socket puis analyser les premiers caractères jusqu'au première «\n», le chiffre ainsi trouvé correspond à la longueur de la trame finale; cela permet de ne pas allez plus loin que la trame et de ne pas endommager une possible information importante.
Le caractères suivant, définit le type de trame (voir le type énumératif «enum_type» du diagramme des classe p 21).
Connaissant le type de trame, il est maintenant aisé de décoder celle-ci :

- Pour les trames de type «i», le pseudo est le texte, avant le délimiteur «\n», puis le mot de passe jusqu'à la fin de la trame.
- Pour les trames de type «n», les 4 premiers octets sont le codage d'un entier long 32 bits, suivis du pseudonyme du joueur, pour l'afficher dans la liste.

3. Diagrammes des classes

1. Classes de la partie Serveur {Olivier LONZI}



EXPLICATION DU DIAGRAMME DES CLASSES LA PAGE PRÉCÉDENTE

Toutes les classes présentes sur ce diagramme sont utilisées par le serveur à l'exception de la classe «*IFCommunicationClient*», qui permet la jonction entre ma partie et celle de mes collègues ayant à dialoguer avec le serveur.

Ce diagramme est une version modifiée, prenant en compte certains ajouts suite au codage; cependant, n'apparaissent pas les attributs ou méthodes propres à un système d'exploitation.

Les quatre classes principales sont celles entourées :

- **IFCommunication** permet la connexion à un socket ainsi que la gestion de l'envoi/réception des trames, par petits paquets; de plus sa méthode ResoudreDNS() permet de trouver une adresse IP. en fonction d'un alias DNS.
- **IFCommunicationGestionTrames** est la classe qui sert de pied d'échange entre le serveur et le client, en simplifiant humainement l'envoi des trames.

  Par exemple le client appellera la méthode «ClientIdentification('pseudo', 'mot_de_passe')» et le serveur recevra dans son thread d'écoute l'information de l'arrivée d'une nouvelle trame d'identification, qui sera rangée automatiquement dans une structure, séparant pseudonyme et mot de passe, en deux variables texte, afin d'être traitée plus facilement.

  Le procédé est le même dans l'autre sens, le serveur utilise des méthodes relativement simples à comprendre d'après leurs noms et le client récupère le tout dans une structure.

  Ladite structure implémente une union, pour éviter l'utilisation d'une trop grande zone mémoire alors que les informations reçues n'utilisent jamais tous les champs existants.

  Attention : La liste des méthodes n'est pas représenté dans sa totalité sur le diagramme mais vous pourrez trouver la documentation complète en annexe page 36.
- **IFCommunicationServeur** est la classe utilisatrice de ses parents, d'un point de vu «serveur»; elle permet d'indiquer le port à utiliser pour l'écoute ainsi que gérer la connexion entre les clients.
- **IFCommunicationClient** permet de lier la communication simple entre le client et le serveur, comme dit plus haut; cette classe est implémentée côté client afin de se connecter à une IP et à un port,:la généralisation avec ses classes parentes permet aux développeurs d'utiliser nativement l'objet de type **IFCommunicationGestionTrames**.

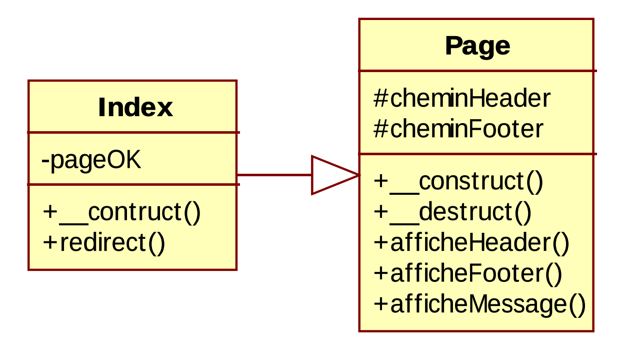
Les deux autres classes présentées sur le diagramme permettent respectivement d'afficher des messages sur la console et de gérer les règles de combat :

- **IFMessages** permet de créer sous forme de file, la liste des messages à afficher sur la console: j'ai découvert l'importance de cette classe pendant le codage et les tests. Le fait est qu'il y a autant de threads que de clients et, sans créer un objet statique, je ne savais pas dans quel sens allaient s'afficher les messages, avec la possibilité que ceux-ci s'intercalent l'un dans l'autre.

- **IFGestionReglesCombats** est une classe qui permet de traiter les combats entre deux joueurs. Après la création de l'objet, rattaché aux threads respectifs des deux joueurs, l'objet de type ***IFCommunicationServeur*** appelle la méthode «*GestionAutomatique*()» des objets de type ***IFGestionReglesCombat***, qui permet de traiter les étapes du combat.

  Attention : Cette classe ne gère pas la complétude des demandes du cahier des charges, quelque soit l'élémentaire utilisé lors des combats, les dégâts sont les mêmes, voir bilan p 30, pour les détails.

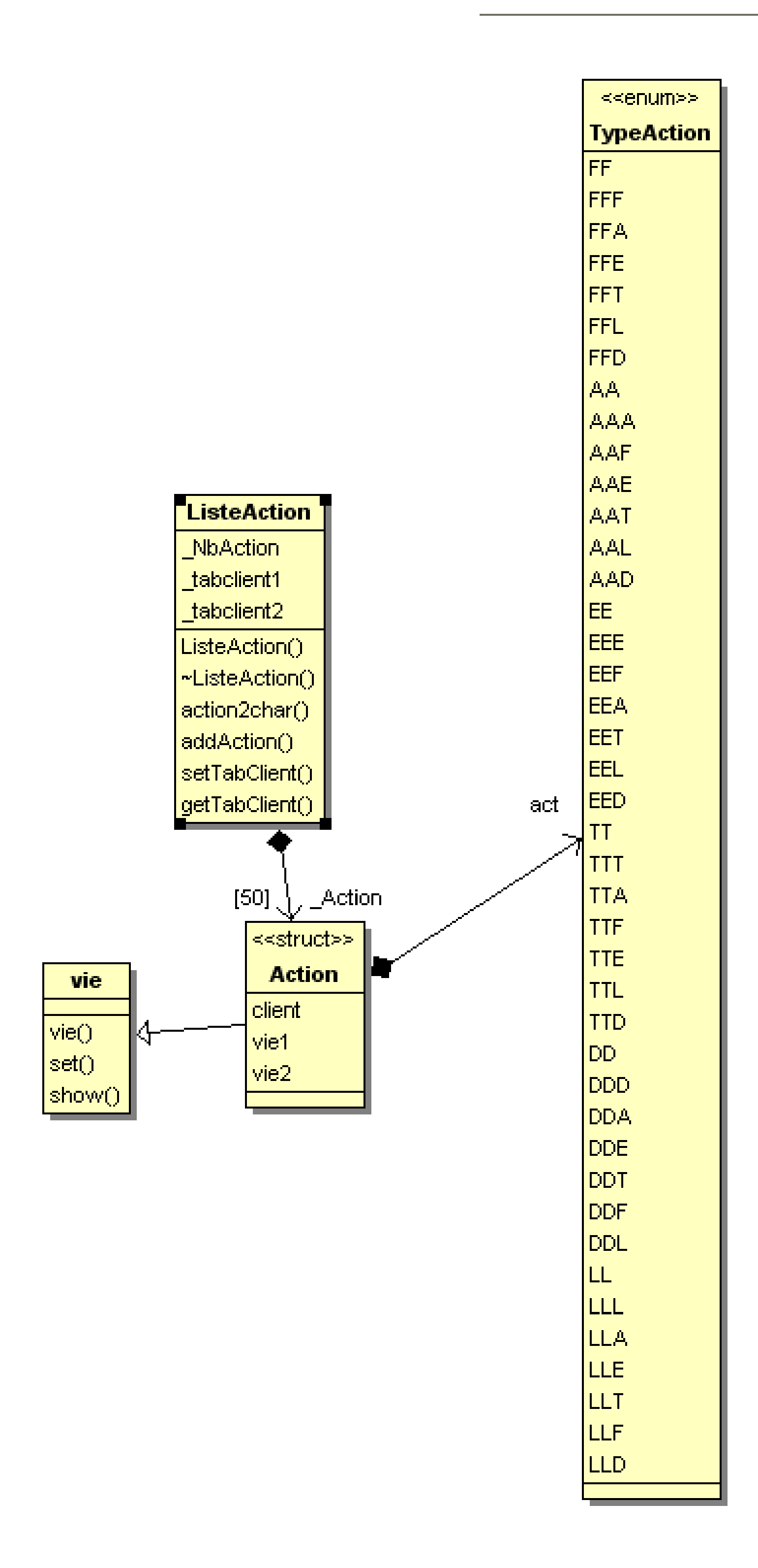
2. Classes de la partie Web {Jean-François BERNARD}



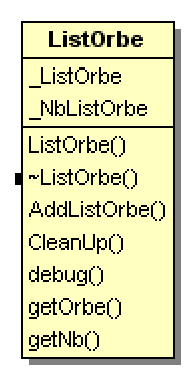
La classe «Page» est composée de deux attributs correspondant aux chemin d'accès de l'entête et du pied de page du site Internet. Elle possède aussi cinq méthodes dont le constructeur et le destructeur ainsi que les méthodes permettant l'affichage de l'entête, du pied de page et aussi des messages aidant à la détection des bugs.

La classe «Index» est composée d'un attribut correspondant aux différentes pages autorisées à être afficher. Elle possède aussi deux méthodes qui sont le constructeur et la méthode affichant les pages autorisées.

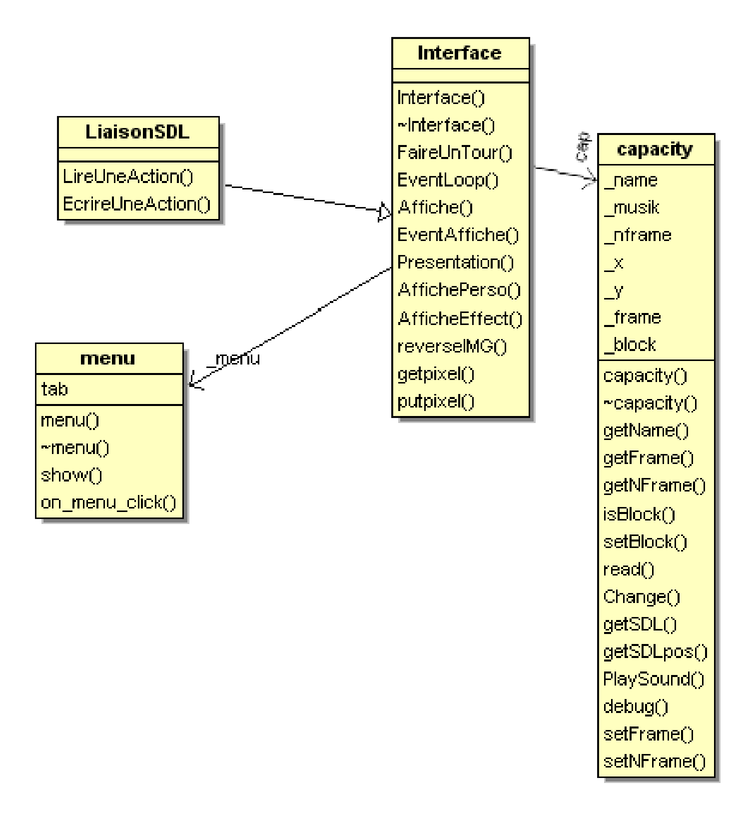
3. Classes de la partie SDL {Jean-Yves MARIN}



Les données correspondant aux actions à effectuer sont ordonnées de cette manière: une liste d'actions ayant les orbes de chaque joueur, les jauges de vie à afficher ainsi que la liste des action faites.



Les données renvoyées au serveur sont une liste des orbes que le joueur a souhaité utiliser.



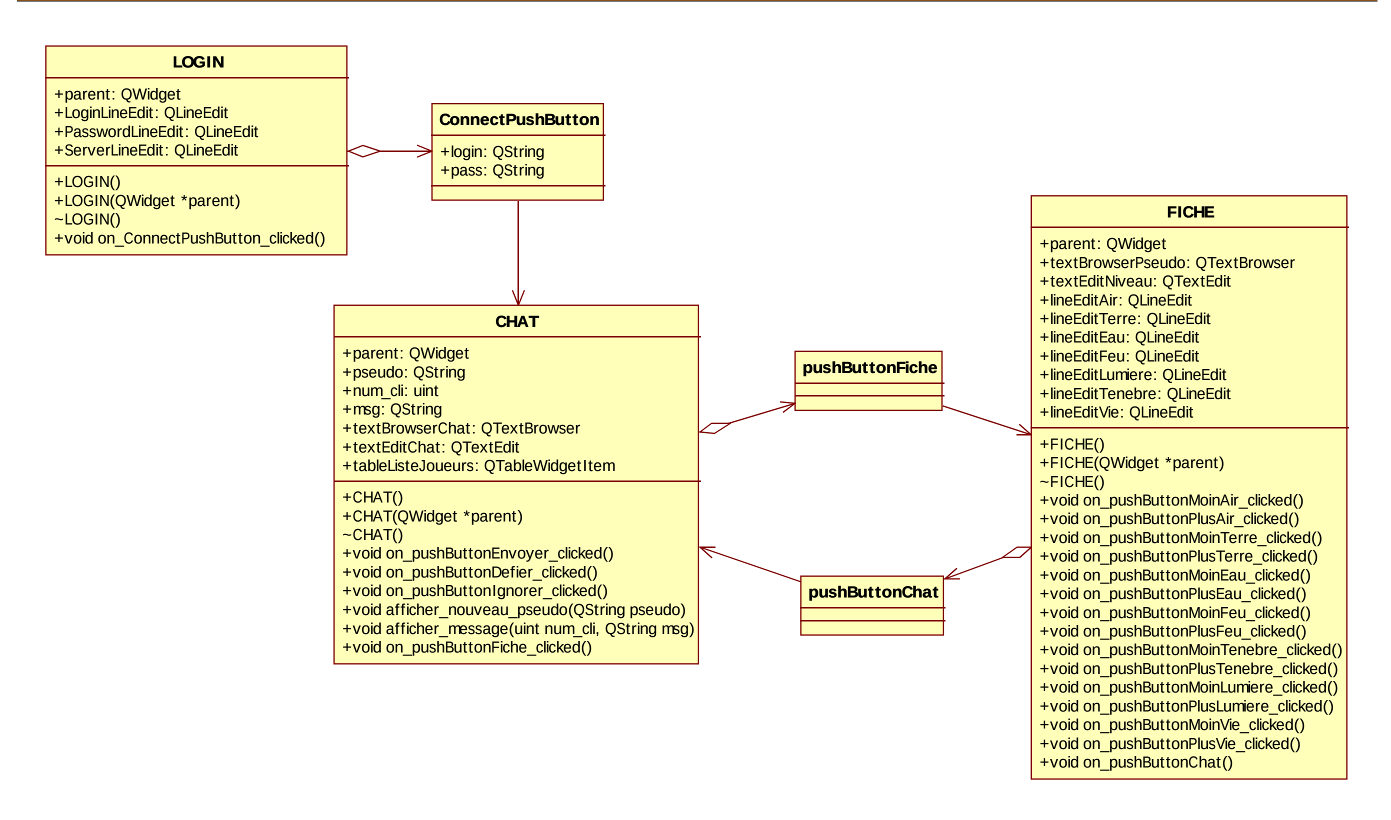
Interface est une classe ayant comme méthode principale « faire un tour » . Elle consomme une liste d'actions et renvoie une liste d'orbes.

Dans un premier temps elle affiche les actions une à une en interrogeant les objets capacity. Puis elle affiche un menu pour récupérer la sélection du joueur.

L'objet capacity sert à lire les fichiers XML et devient une liste de chemins pouvant être comprise pour l'affichage. Le nom est bien mal accordé avec sa fonction, « animation » serai meilleur. Les règles du jeux ayant évoluées le code n'a pas suivi en tout point.

La liaison SDL serre uniquement à intégrer la partie réseau et rendre transparente les mécanismes de la partie interface, c'est pour cela qu'elle ne comporte que deux fonctions.

4. Classes de l'interface client {Édouard BOUCHER}



Les 3 classes principales sont les classes de "LOGIN", de "CHAT" et de "FICHE".

La classe "LOGIN" permet grâce au signal "on_ConnectPushButton_clicked", d'accéder à la fenêtre de chat en prenant en compte les paramètres de login et de password.

La classe "CHAT" permet grâce aux fonctions "afficher_nouveau_pseudo" et "afficher_message" d'afficher la liste des personnes connectées au serveur et d'afficher des messages des utilisateurs. Enfin, elle permet grâce aux signaux on_pushButtonEnvoyer_clicked", "on_pushButtonDefier_clicked", "on_pushButtonIgnorer_clicked" et "on_pushButtonFiche_clicked" d'envoyer un message au chat, de défier ou ignorer le défi d'un joueur, et d'accéder à la fiche de son personnage.

La classe "FICHE" permet d'afficher le pseudo du joueur, son niveau et les points de compétences en cours aux moyens des différents attributs textBrowerPseudo, textEditNiveau et lineEdit(Air, Terre, Eau...).

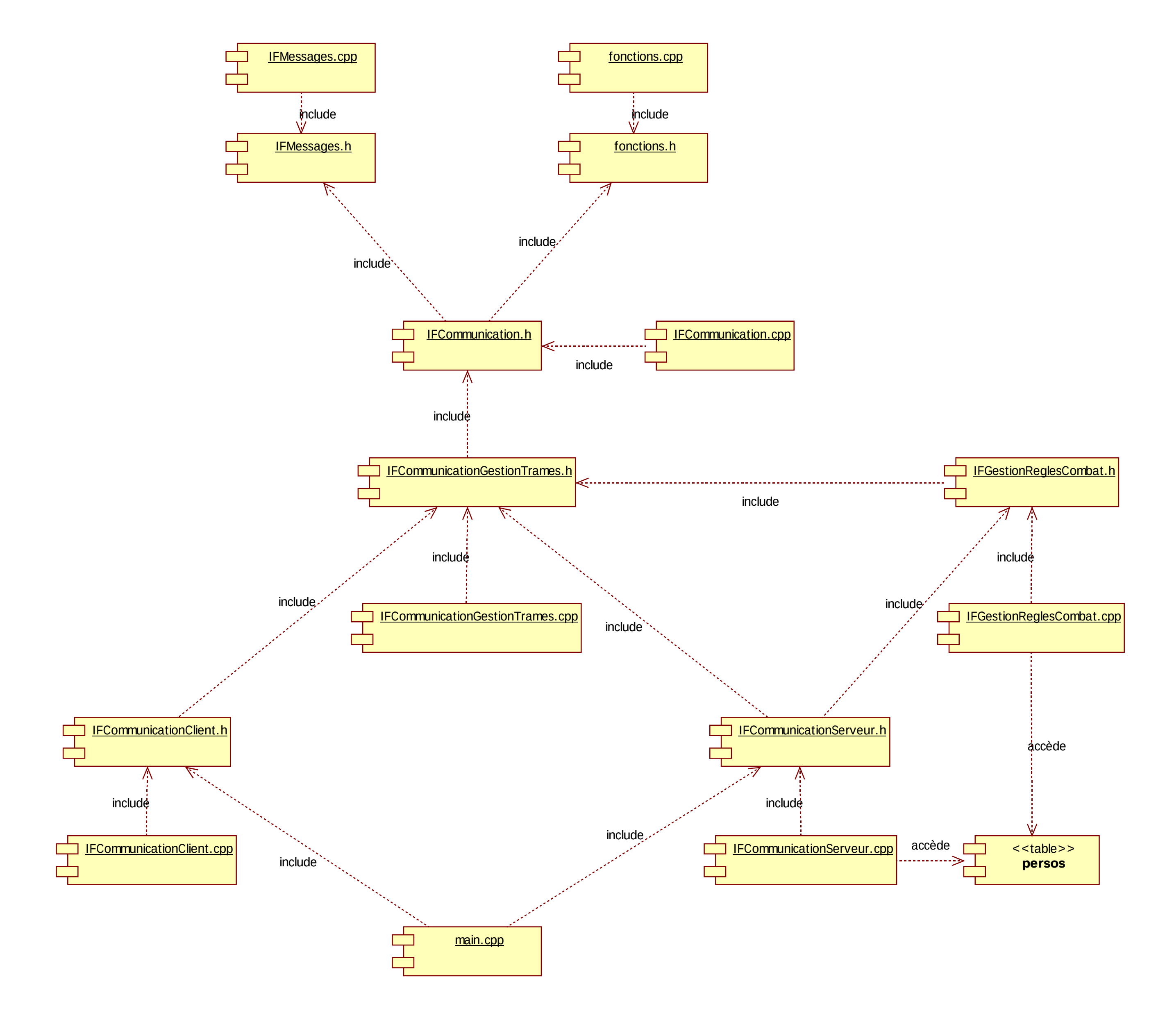
Elle permet également grâce à ses méthodes pushButtonMoin(Air, Terre ...) et pushButtonPlus(Air, Terre ...) de modifier les compétences des différents orbes en cas de passage de niveau.

Enfin, le signal on_pushButtonChat permet d'accéder à la fenêtre de chat.

4. Diagrammes des composants

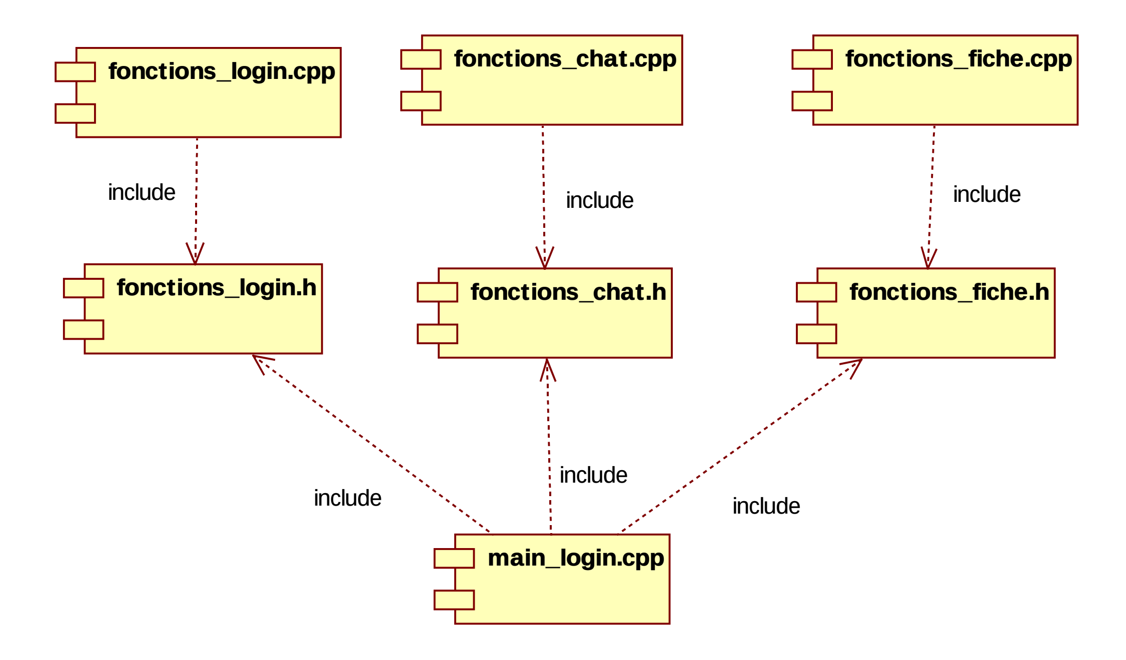
Le diagramme des composants permet de modéliser un ensemble d'objets logiciels comme les fichiers ou les tables contenues dans des bases de données.

1. Composants de la partie Serveur {Olivier LONZI}



Dans ce diagramme, nous pouvons voir les différents fichiers sources utilisés ainsi que les fichiers qui ont besoin d'accéder à certaines tables MySQL.

2. Composants de la partie interface graphique {Édouard BOUCHER}



Le fichier main_login.cpp est le fichier principal incluant les headers des fenêtres de login, de chat et de fiche personnage. Les fichiers fonctions_login.cpp, fonctions_chat.cpp et fonctions_fiche.cpp regroupent les fonctions propres aux fenêtres correspondantes.

IX. Conception détaillée

1. Plan de Tests Unitaires

1. Partie Serveur {Olivier LONZI}

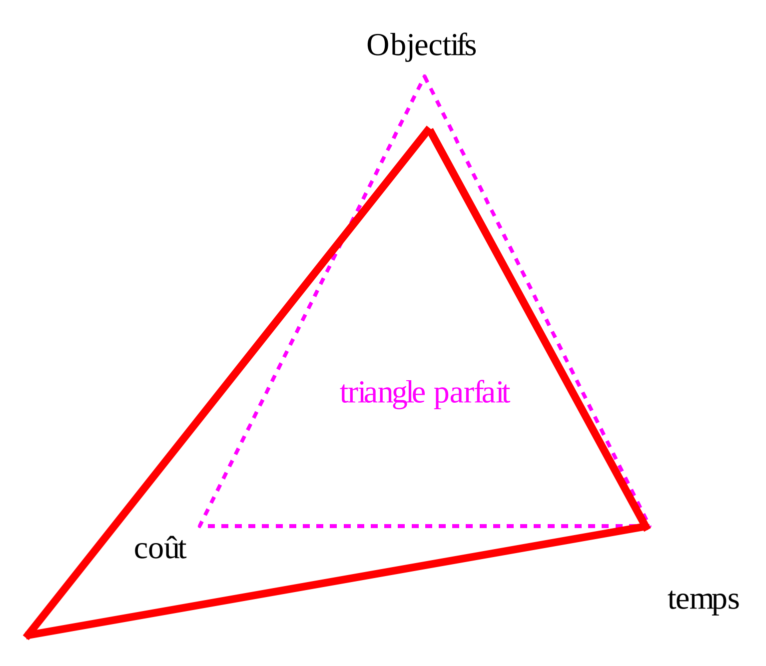
Actions à vérifier	Tests effectués	Résultats attendus
Écoute des clients	Lancer un client test sur le l'IP et le port.	Annonce d'une nouvelle connexion sur le serveur.
	Lancer deux clients sur l'IP et le port.	Annonce d'une nouvelle connexion, puis d'une seconde.
Identification d'un client	Login : test – mdp : faux	Refuser la connexion
	Login : Miranda – mdp : a	Accepter la connexion
Demande de liste des clients connectés	1 seul client connecté	Envoyer le numéro du client et son pseudo.
	2 clients connectés	Envoyer un par un le couple (numéro de client et pseudo) pour chacun des deux clients.
Envoyer un message (3 clients connectés)	Le joueur 1 envoie le message «bonjour» à tout le monde	Le serveur reçoit le message et le retransmet à tout le monde, y compris l'expéditeur, en indiquant le numéro de J1 (0).
	Le joueur 3 envoie le message «salut» à tout le monde	Le serveur reçoit le message et le retransmet à tout le monde, y compris l'expéditeur, en indiquant le numéro de J3 (2).

Vous trouverez les tests dans l'annexe p 41.

X. Bilan

1. Triangles des projets

Comme demandé dans le document fourni pas M. JAGOURY (Professeur de Gestion de projet), la notion de coûts sera adaptée en comparant la quantité de travail fournie par rapport à la quantité de travail théorique affecté au projet.



2. Explication du triangle

Comme le montre le triangle du projet, le système final ne correspond pas parfaitement aux exigences du cahier des charges, ceci est dû à la non gestion des règles de combat exactement comme l'exigeait l'analyse. Un système plus rudimentaire a été mis en place, des points de dégât sont attribués suivant le niveau du personnage et de ses affinités avec l'élément utilisé, mais tous les effets détaillés dans l'annexe p 38, ne sont pas fonctionnels. Ce retard s'explique par des problèmes dus à l'intégration et à la définition des règles après le cahier des charges et les analyses du projet.

Les temps sont quant-à eux respectés, car le projet était fondé autour d'une date limite, nous ne pouvions donc pas prendre plus de temps. Le cahier des charges n'étant pas rempli, nous ne pouvions pas annoncer une fin prématurée.

En ce qui concerne les coûts (quantité de travail fournie / quantité de travail affecté), aucune affectation de travail n'a été réalisé lors de l'analyse, car nous n'avions pas fait le cours correspondant, et aucun d'entre nous n'avait de connaissance de ce système. Cependant, il est certain que nous avons tous eu plus de travail à réaliser, ne serais-ce que pour la communication entre toutes les parties: cela explique l'explosion du triangle ci-dessus, dans la partie des coûts.

XI. Résumer du projet

1. En français

IRIS FIGHT est un jeu en ligne mettant en scène deux joueurs dans un combat tour par tour. IRIS FIGHT permet aussi la communication entre les joueurs et dispose d'une interface de compte en ligne mis en place sur un site Internet. Ce jeu propose la création d'un personnage qui au fur et à mesure des combats gagne de l'expérience que le joueur pourra dépenser dans ses différentes capacités.

2. En anglais

IRIS FIGHT is an online turn-based battle game between two players. IRIS FIGHT also allow the communication between players and have an online account's interface on his website. This game purpose a character creation that gain experience points at the end of the battle. This point can be spend in his different capacities.

XII. Conclusions personnelles des étudiants

1. Jean-Yves MARIN

Le projet tutoré m'a apporté une expérience d'équipe avec des membres dont je ne connaissais pas les acquis. Je constate que si l'on reprenait ce travail au début, l'on travaillerait différemment. Du point de vue de la programmation, j'ai découvert la bibliothèque SDL et la bibliothèque FMOD. Le projet m'a aussi permis de travailler avec du XML.

J'ai pu apporter mon soutient et mes connaissances pour la conception des Makefile et des fichiers projet Qt.

J'ai limité au maximum les méthodes de mon API pour n'avoir à utilisé que deux actions: Exécuter une liste d'action et renvoyer une liste d'action de l'utilisateur.

J'ai pu aussi utiliser le programme doxygene pour pouvoir fournir à mes équipiers une documentation simple.

2. Édouard BOUCHER

Travailler sur ce projet fut gratifiant pour moi sur plusieurs points.

Sur le plan humain, j'ai pu apprendre ce que c'est de travailler en groupe sur un projet important. Notre projet nous obligeant à avoir des rôles étroitement liés afin de combiner nos sources et avoir un exécutable unique, la communication et l'organisation étaient plus que primordiales sur ce projet.

Sur le plan informatique et dans le cadre de ma formation, cela m'a permis de renforcer grandement mes connaissances en Qt, une librairie que je connaissais peu et que j'ai pris plaisir à découvrir plus en profondeur. J'ai également renforcé mes connaissances en C++ afin de faire des sources les plus simples possibles, pour que mes camarades puissent comprendre facilement comment elles sont structurées.

Tout ces points seront certainement une expérience positive pour mon futur au cours de mon futur parcours professionnel et m'ont montré comment m'intégrer et travailler au sein d'une équipe.

3. Olivier LONZI

J'avais déjà expérimenté le travail en groupe, au cours de mon année de BTS IRIS 2007, durant laquelle nous avions dû programmer un système d'arrosage intelligent dans le but d'être mis en place dans une mairie. À l'instar de cette année, où le cahier des charges du projet a été réalisé par nous même, pour nous même ; j'ai donc pu constater que ce genre de projet est beaucoup moins motivant que lorsque le travail à fournir à un réel but d'utilisation pour une entreprise. Cependant je compte bien mettre en place un serveur de jeu toujours connecté en ligne.

J'ai profité de ce projet pour approfondir mes connaissances dans l'utilisation des sockets et surtout dans la programmation sous Linux, en fixant délibérément la portabilité du serveur sous Windows et Linux dans le cahier des charges.

J'ai pu aussi apprendre, lors du travail en groupe, à utiliser un peu Qt.

Malgrès quelques mise en garde vis-à-vis de la création d'un jeu plutôt que d'un utilitaire logiciel, je pense que pour nos projets futures, la méthodologie de travail en groupe, et les nouveaux acquis, sont des points gratifiants pour la suite de nos cursus.

4. Jean-François BERNARD

Suite à l'arrêt de mon projet tutoré initial, j'ai pris plaisir à rejoindre ce groupe qui a accepté que je me joigne à eux pour le développement de ce projet. Sachant qu'ils avaient déjà effectué le partage des différentes tâches de ce projet, j'ai proposé de m'y intégrer par l'intermédiaire d'un site Internet permettant de gérer la création et l'évolution du personnage d'un joueur.

D'un point de vue humain, j'ai apprécié travailler dans ce groupe qui m'a bien expliqué les principes et but de leur projet lors de mon arrivée. De plus, le soutien et l'aide qu'ils m'ont apporté lors de la création du site Internet m'a permis de le construire dans un but s'intégrant parfaitement dans les objectifs de ce projet.

D'un point de vue informatique, j'ai pu approfondir mes connaissances dans les différents langages que j'ai utilisé (HTML, CSS, PHP, MySQL). De plus, mes connaissances en gestion de serveur et hébergement de site web se sont enrichies grâce aux discussions que j'ai pu avoir avec les membres de mon groupe à propos de l'intégration fonctionnelle de mon site Internet vis à vis de ce projet.

Pour conclure, je considère que mon intégration à ce projet a été des plus réussi, qu'elle m'a permis d'apprécier le travail en groupe ainsi que d'approfondir mes connaissances en développement Internet.

XIII. Modifications à apporter au programme dans le futur

Dans le futur, il serait dans un premier temps envisageable de terminer complètement la gestion des règles.

Par la suite nous pourrions envisager d'effectuer les améliorations suivantes :

- Gestion d'une intelligence artificielle.
- Apporter une finition au graphisme.
- Éditeur d'animation.
- Et sûrement plein d'autres choses euphoriques :-).

XIV. Annexes

1. Sources de références

http://www.developpez.com/

De bonne références et des exemples clairs d'algorithmes, sur différents langages de programmation.

http://tangentsoft.net/mysql++/

Site de la librairie officielle pour lier les applications c++ à MySQL.

Documentation MANuel Linux

La documentation MAN donne toutes les informations nécessaires sur les fonctions C/C++

http://www.php.net/

Site donnant toutes les informations nécessaires sur les fonctions PHP.

http://www.selfhtml.fr

Site expliquant la plupart des fonctionnalités HTML et CSS.

2. Liste des méthodes d'échange entre le client et le serveur

Toutes les méthodes listées sont utilisables par un objet de type IFCommunicationClient().

IFCommunicationGestionTrames()

> Le constructeur par défaut.

IFCommunicationGestionTrames (SOCKET, SOCKADDR_IN);

> Ce constructeur permet d'instancier un objet de gestion des trames avec les informations d'un socket déjà ouvert.

~IFCommunicationGestionTrames ();

> Le destructeur de l'objet.

int EnvoyerMessageSalon (const unsigned char *);

> Cette méthode permet au client d'envoyer un message à tous les autres joueurs.

int EnvoyerMessageClient (unsigned long num_client_dst, const unsigned char *);

> Le client envoie un message à un autre client. Ce message n'est pas envoyé à tout le monde.

int DemanderListeClients ();

> Fait la demande au serveur, pour connaitre la liste des clients actuellement connectés. En réponse le serveur enverra une liste de «Connexion client». **Attention :** Les nouveaux clients qui rejoindront le serveur par la suite seront automatiquement notifiés aux clients déjà présents.

int QuitterServeur ();

> Le client informe qu'il souhaite quitter le serveur.
>
> Tous les clients encore connectés reçoivent une notification de déconnexion.

int ProposerDuel (unsigned long num_client);

> Un client propose un duel avec un autre client, le serveur envoie une notification à ce second client.
>
> Attention : Un client ne peut envoyer qu'une demande de duel à la fois.

int ClientIdentification (const unsigned char * pseudo, const unsigned char * mdp);

> Le client s'identifie auprès du serveur. En réponse le serveur envoie en retour un code d'acceptation ou de refus.

int AccepterDuel (unsigned long num_client);

> Informe le serveur qu'un duel est accepté, termine automatiquement les demandes de duel en cours. Il faut préciser le nom du client qui avait lancé le défi.

int RefuserDuel (unsigned long num_client);

> Informe du refus d'un duel.

int GererReceptionRequete (struct_reception_donnee &);

> Cette méthode permet de gérer les retours au travers d'une structure.
>
> (voir exemple sur le diagramme des classes p 21)

3. Règles de combat {Jean-Yves MARIN}

1. Introduction

IRIS FIGHT est un jeu où s'affrontent 2 personnes.
Le combat se déroule en tour par tour, pendant lequel les deux joueurs lancent leurs sorts dans un ordre précis.

2. Déroulement d'un match

Les deux joueur choisissent 3 orbes qu'ils auront pour débuter le combat.
Il y a 6 types d'orbes : Air, Terre, Feu, Eau, Lumière, Ténèbres.

Les deux joueurs choisissent un orbe qu'ils vont invoquer pour le rajouter à leurs ressources, puis un autre leur est donné de façon aléatoire.
Ensuite, les joueurs choisissent de faire, ou non, un ou plusieurs sorts, ce qui consomme des orbes.
Les sorts sont ensuite lancés dans un certain ordre: les défenses, les défenses offensives, et les attaques.
Si plusieurs sorts ont la même priorité, on détermine de manière aléatoire qui le lancera en premier.
Les dégâts sont infligés.

3. Principe de base de l'alchimie des orbes

Pour être plus intuitif les éléments d'orbes ont un domaine et un effet secondaire.

Légende :

A = Air	*F = Feu*
T = Terre	*L = Lumière*
E = Eau	*D = Ténèbres (darkness)*

Deux orbes d'un même élément font une protection contre cet élément.

Trois orbes d'un même élément font une attaque basique de cet élément.

Exemple:

- Le joueur 1 décide de lancer un sort de feu (FFF) et une protection contre l'eau (EE)
- Le joueur 2 décide de faire un sort d'air (AAA) et un protection au feu (FF).
- Les sorts se lancent toujours dans le même ordre : sort de défense, sort de défense offensive, sort offensif.
- Les protections des deux joueurs se lancent, aléatoirement. J1 est protégé contre l'eau, J2 contre le feu.
- Il n'y a pas de défense offensive. On passe au sort d'attaque.
- J1 n'inflige pas de dégâts car J2 avait bien anticipé, mais J2 inflige bien les dégâts.
- Les défenses ne durent que pendant le tour courant.

Deux orbes de même type et un orbe d'un autre élément lanceront un sort du type de l'orbe unique, avec les caractéristiques de domaine des orbes majoritaires.

Example.
Le feu a comme domaine de la puissance (plus de dégâts avec l'élément associé) mais pas d'effet secondaire.
Donc si je fais FFT je lance un attaque d'élément terre , ne faisant pas d'effet secondaire mais infligeant plus de dégâts.

Note: TTF, TFT, et FTT sont les mêmes sorts.

4. Domaines et effets secondaires

Il y a 4 niveaux de dégâts : aucun, faible, moyen, lourd.
Un sort basique (ex: TTT) fait des dégâts moyens.

Air :
Effet secondaire : Shock : 20% de chance par attaque (pendant 1 tour par niveau d'attaque (faible=1,moyen=2...), l'adversaire a 20% de chances de rater ses sorts).
Domaine : Effet secondaire (augmente les chances d'infliger des effets secondaires, mais diminue les dégâts)

Terre :
Effet secondaire : Assommoir : 10% de chances d'annuler les N prochains tours de l'adversaire. N est défini par le niveau de l'attaque.
Domaine : Protection offensive.(inflige des dégâts faibles du type d'élément associé, défense contre l'élément associé)

Eau :
Effet secondaire : Chance de geler (pareil que assommoir)
Domaine : Vie (soin faible, défense contre l'élément associé)

Feu :
Effet secondaire : Brulure : 10% de chances (inflige 100% des dégâts initiaux en plus, répartis sur 3 tours.)
Domaine : Puissance (Attaque lourde de l'élément associée mais sans effet secondaire).

Ténèbre :
Effet secondaire : Vampirisme 25% (récupère 25% des dégâts infligés).
Domaine : Faiblesse (affaiblie la cible à l'élément associé, les dégâts sont multipliés par 1,5 pendant 3 tours pour l'élément associé)

Lumière :
Effet : Destruction d'orbes 15% (détruit N orbes aléatoires de l'adversaire, N étant défini par le niveau de l'attaque.)
Domaine : Vol d'orbes (vol 2 orbes de l'élément associé)

5. Les pentacles

Cinq orbes de même élément invoque le pentacle de l'élément, sort ultime de l'élément.
(A découvrir....).

6. Liste des sort

Sorts d'air :

AA : Défense contre l'air.
AAA : Attaque moyenne d'air.

FFA : Eclair flamboyant (dégât lourd, pas d'effet secondaire)
EEA : Corps vaporeux (défense contre l'air, soin faible)
TTA : Mur de vent (dégât faible, défense contre l'air)
DDA : Affaiblissement à l'air (peut être contré pas une défense contre l'air)
LLA : Implosion (vol 2 orbes d'air, peut être contré pas une défense contre l'air)

AAAAA : Pentacle d'air.

Sorts d'eau :

EE : Défense contre l'eau.
EEE : Attaque moyenne d'eau.

FFE : Vapeur ardente (dégât lourd d'eau, pas d'effet secondaire)

AAE : Brouillard givrant (dégât faible, double les chances d'effet secondaire)
TTE : Mur de glace (dégât faible, défense contre l'eau)
DDE : Affaiblissement à l'eau (peut être contré pas une défense contre l'eau)
LLE : Archimede (vol 2 orbes d'air, peut être contré pas une défense contre l'eau)

EEEEE : Pentacle d'eau.

Sort de terre :

TT : Défense contre la terre.
TTT: Attaque moyenne de la terre.

FFT : Lance ardente (dégât lourd de terre, pas d'effet secondaire)
AAT : Pierre guidée (dégât faible, double les chances d'effet secondaire)
EET : Corps d'argile (soin faible, défense contre la terre)
DDT : Affaiblissement à la terre (peut être contré pas une défense contre la terre)
LLT : Aimant (vol 2 orbes de terre, peut être contré pas une défense contre la terre)

TTTTT : Pentacle de terre.

Sort de feu

FF : Défense contre le feu.
FFF: Attaque moyenne de feu.

TTF : Mur de feu (dégât faible de feu, défense contre le feu)
AAF : Brasier attisé (dégât faible, double les chances d'effet secondaire)
EEF : Corps d'huile ardente (soin faible, défense contre le feu)
DDF : Affaiblissement au feu (peut être contré pas une défense contre le feu)
LLF : Feu follet (vol 2 orbes de feu, peut être contré pas une défense contre feu)

FFFFF : Pentacle de feu.

Sort de lumière

LL : Défense contre la lumiere.
LLL: Attaque moyenne de lumiere.

TTL : Mur aveuglant (dégât faible de lumière, défense contre la lumière)
AAL : Mirage (dégât faible, double les chances d'effet secondaire)
EEL : Corps luisant (soin faible, défense contre la lumière)
DDL : Affaiblissement à la lumière (peut être contré pas une défense contre la lumière)
FFL : Laser (dégât lourd, pas d'effet secondaire)

LLLLL : Pentacle de lumiere.

Sort des ténèbres

DD : Défense contre les ténèbres.
DDD: Attaque moyenne des ténèbres.

TTD : Mur des abysses (dégât faible des ténèbres, défense contre les ténèbres)
AAD : Voile des tenebres (dégât faible, double les chances d'effet secondaire)
EED : Corps noir (soin faible, défense contre les ténèbres)
LLD : Clair obscure (vol 2 orbes de ténèbres, peut être contré pas une défense contre les ténèbres)
FFD : Flamme noire (dégât lourd, pas d'effet secondaire)

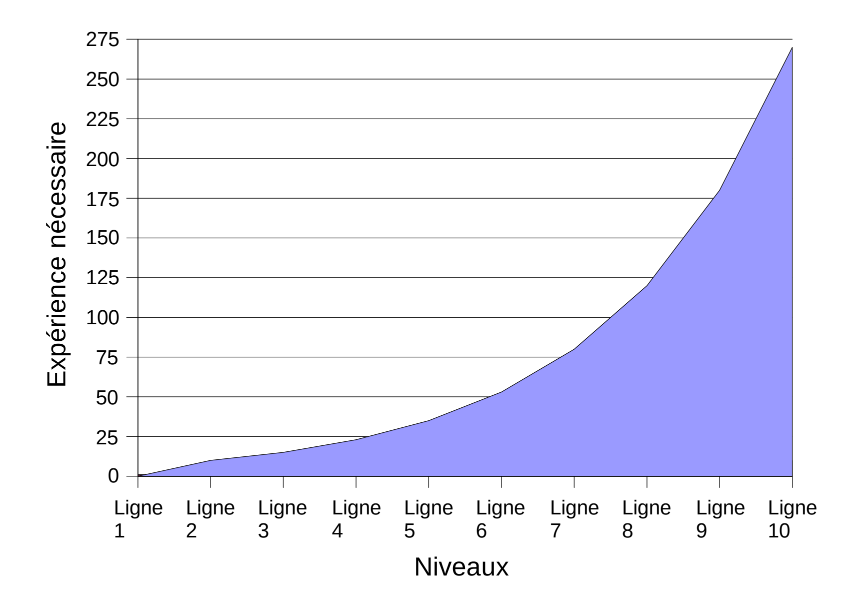
1. DDDDD : Pentacle des ténèbres.

7. Expérience

Après un certaine nombre de combat ,le joueur peut changer de niveau, il peut gagner 3 point à mettre dans C caractéristique
les caractéristiques sont les 6 éléments et la vie
Chaque point augmente de 10% les effet des sorts de l'élément associé, de plus les défenses absorbent le mêm pourcentage lors d'une attaque au lieu de simplement les arrêter.
La vie n'augmente que de 3% par point de compétence utilisé.

Les changements de niveau se font suivant la règles suivantes :

- Passage au niveau 2 : 10 points d'expérience
- Passage au niveau 3 : 15 points d'expérience {10 * 1,5}
- Passage au niveau 4 : 23 points d'expérience {15 * 1,5}
- Passage au niveau 5 : 35 points d'expérience {23 * 1,5}
- et ainsi de suite.



Les joueurs terminant un combat morts gagne 1 point d'expérience, les joueurs remportant le match gagne 3 points d'expérience.
Les joueurs se déconnectant d'un match ne gagne aucun point d'expérience, le joueur restant gagne 2 points d'expérience.

8. Dégât

- Faible : 1D4
- Moyen : 1D4 + 3
- Fort : 2D4 + 2

4. Tests Unitaires

1. Partie Serveur {Olivier LONZI}

Pour illustrer les résultats j'effectue des impressions écran de la console dos, en inversant les couleurs pour éviter d'avoir trop de noir lors de l'impression. Les résultats obtenus sont strictement identiques à ceux du système d'exploitation Linux, mis à part l'indicateur d'état en haut à droite, qui permet de savoir très approximativement la vitesse de bouclage du serveur, et qui n'est pas implémentable sous Linux (pas de moyen de récupérer la position du curseur).

1. Écoute des clients

1. Lancer un client test sur le l'ip et le port.

```
D:\WINDOWS\system32\cmd.exe - IFServeur.exe
[OK]    Instanciation d'un IFCommunication (1)
[OK]    Creation du socket [*:1664]
[OK]    Creation du serveur (BIND) [*:1664]                    Etat : \
[OK]    Ecoute sur l'entree du serveur (LISTEN) [*:1664]
[OK]    Client accepte (ACCEPT)
[OK]    Instanciation d'un IFCommunication (2)
[OK]    Creation d'un thread
```

Impr. écran 1 :Connexion d'un client non identifié

Un client se connecte au serveur, celui-ci affiche la connexion, comme spécifié dans les résultats attendus du plan de test.

2. Lancer deux clients sur l'IP et le port.

```
D:\WINDOWS\system32\cmd.exe - IFServeur.exe
[OK]    Instanciation d'un IFCommunication (1)
[OK]    Creation du socket [*:1664]
[OK]    Creation du serveur (BIND) [*:1664]                    Etat : \
[OK]    Ecoute sur l'entree du serveur (LISTEN) [*:1664]
[OK]    Client accepte (ACCEPT)
[OK]    Instanciation d'un IFCommunication (2)
[OK]    Creation d'un thread
[OK]    Client accepte (ACCEPT)
[OK]    Instanciation d'un IFCommunication (3)
[OK]    Creation d'un thread
```

Impr. écran 2 :Connexion de deux client non-identifiés

Dans le cas où deux clients se connectent le serveur affiche les deux connections l'une après l'autre comme spécifié dans le plan de test.

2. Identification d'un client

1. Login : test – mdp : faux

```
D:\WINDOWS\system32\cmd.exe - IFServeur.exe
[OK]     Instanciation d'un IFCommunication (1)
[OK]     Creation du socket [*:1664]
[OK]     Creation du serveur (BIND) [*:1664]                     Etat : /
[OK]     Ecoute sur l'entree du serveur (LISTEN) [*:1664]
[OK]     Client accepte (ACCEPT)
[OK]     Instanciation d'un IFCommunication (2)
[OK]     Creation d'un thread

              ->Trame recus : "itest
faux"
Thread Client STOP
[OK]     Envoie de la trame "1\nr" sur [1896]
[OK]          Client 0 (Fermeture du thread)
```

Impr. écran 3 :Tentative d'identification échouée

Le serveur reçoit la trame d'identification «itest\nfaux» (voir décodage des trames p 20). Ce compte n'existe pas, ou le mot de passe est faux: le serveur envoie donc une trame de refus «r» au client, puis ferme le thread ouvert.

2. Login : Miranda – mdp : a

```
[OK]     Client accepte (ACCEPT)
[OK]     Instanciation d'un IFCommunication (3)
[OK]     Creation d'un thread

              ->Trame recus : "iMiranda
a"
[OK]     Envoie de la trame "1\na" sur [1864]
```

Impr. écran 4 :Tentative d'identification réussie

Le serveur reçoit la trame d'identification «iMiranda\na» (voir décodage des trames p 20). Ce compte existe, de plus le mot de passe est juste: le serveur envoie donc une trame d'acceptation «a» au client.

3. Demande de liste des clients connectés

1. Un seul client connecté

```
              ->Trame recus : "l"
[OK]     Envoie de la trame "12\nn____Miranda" sur [1864]
```

Impr. écran 5 :Demande liste des clients (1 seul client)

Le joueur demandant la liste reçoit le seul numéro (0) et pseudo. Donc le sien.

2. Deux clients connectés

```
              ->Trame recus : "l"
[OK]     Envoie de la trame "12\nn____Miranda" sur [1768]
[OK]     Envoie de la trame "15\nn___☺Jean-Louis" sur [1768]
```

Impr. écran 6 :Demande liste des clients (2 clients)

Le joueur demandant la liste reçoit les deux couples (numéro, pseudo), il peut en déduire son numéro suivant son pseudo.

4. Envoyer un message (3 clients connectés)

1. Le joueur 1 envoie le message «bonjour» à tous le monde

```
                ->Trame recus : "sbonjour"
[OK]    Envoie de la trame "12\ns____bonjour" sur [1896]
[OK]    Envoie de la trame "12\ns____bonjour" sur [1780]
[OK]    Envoie de la trame "12\ns____bonjour" sur [1732]
```

Impr. écran 7 :Envoie de message «bonjour»

Le joueur 1 identifié par le numéro 0 «____» envoie un message à tout le monde «s», le serveur le reçoit et le retransmet à tout le monde (y compris le joueur ayant envoyé le message), en indiquant le numéro de l'expéditeur.

2. Le joueur 3 envoie le message «salut» à tout le monde

```
                ->Trame recus : "ssalut"
[OK]    Envoie de la trame "10\ns___☻salut" sur [1896]
[OK]    Envoie de la trame "10\ns___☻salut" sur [1780]
[OK]    Envoie de la trame "10\ns___☻salut" sur [1732]
```

Impr. écran 8 :Envoie de message «salut»

Le joueur 3 identifié par le numéro 2 «___☻» (☻ = représentation graphique du caractères 2 ASCII) envoie un message à tout le monde «s», le serveur le reçoit et le retransmet à tout le monde (y compris le joueur ayant envoyé le message), en indiquant le numéro de l'expéditeur.

5. Conclusions sur les TU

Ces tests permettent de déceler divers bogues (minimes dans certains cas), mais parfois beaucoup plus important, surtout une fois l'ensemble intégré.

Pour exemple, la connexion au serveur fonctionnait parfaitement bien, la déconnexion de même, mais lorsque qu'un même nom d'utilisateur s'identifiait juste après une déconnexion, le serveur pouvait le refuser, pour doublon.

Explication :

- Le serveur stocke la liste des pseudos dans un tableau.
- Lors de l'identification de connexion, le serveur vérifie que le pseudo n'et pas dans la liste.
- Dans un premier temps, aucun pseudo, donc pas de problème, mais une erreur survenait, lors de la déconnexion où, bien qu'un ***delete*** indiquait la mémoire comme inutilisée, le pseudo restait en mémoire résidente.
- De ce fait si le même compte tentait de se reconnecter, le serveur lisait la mémoire résidente et refusait le joueur.

Correction :

- Forcer à NULL, la case du tableau correspondant au pseudo, après avoir effectué le ***delete***.

5. Choix des ressources techniques

1. Présentation de la SDL (Simple Directmedia Layer) {Jean-Yves MARIN}



1. Présentation

Le but de la partie SDL est de pouvoir afficher de façon conviviale la partie en cour.
Cette partie n'a qu'un but d'interface, le minimum de données relatives à la partie y est traité.
L'interface affiche l'action donnée par le serveur puis récupère les actions du joueur.

Initialisation

Non

Lecture des actions

Si victoire

Oui

Affichage

Afficher résultat

Récupération des actions du joueur

Rendre la main au chat

2. Avantages et inconvénients

La librairie SDL est une librairie simple et bas niveau. Elle permet aussi de gérer les entrées clavier et souris. Il devient facile de pouvoir mouvoir des personnages si l'on a des fichiers image à disposition mais en revanche il est beaucoup plus difficile de faire un menu. Elle reste bas niveau et écrite en C, non en C++. Les images doivent être construites une à une en plaçant les sprites dans le bon ordre pour ne pas qu'ils se masquent les uns les autres.

3. Utilisation de XML

L'interface utilise la bibliothèque graphique SDL pour afficher image par image le déroulement de l'action. Elle puise les information nécessaires aux animations dans des fichiers XML pouvant par la suite être modifiés. Le fichier XML contient le nombre d'images à afficher pour l'animation, le chemin de l'image, la couleur de transparence et le son associé. Cela permet à l'utilisateur de changer le skin de son personnage ou des actions sans recompilation.

Exemple de fichier XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<action nb="36" r="242" g="0" b="242" sound="ddd">
      <bmp x="356" y="232">../ressource/sprite/effect/ddd/1.bmp</bmp>
      <bmp x="356" y="233">../ressource/sprite/effect/ddd/2.bmp</bmp>
      <bmp x="354" y="232">../ressource/sprite/effect/ddd/3.bmp</bmp>
      <bmp x="354" y="231">../ressource/sprite/effect/ddd/4.bmp</bmp>
      <bmp x="354" y="231">../ressource/sprite/effect/ddd/5.bmp</bmp>
      ...
      <bmp x="430" y="308">../ressource/sprite/effect/ddd/32.bmp</bmp>
      <bmp x="434" y="310">../ressource/sprite/effect/ddd/33.bmp</bmp>
      <bmp x="436" y="313">../ressource/sprite/effect/ddd/34.bmp</bmp>
      <bmp x="440" y="316">../ressource/sprite/effect/ddd/35.bmp</bmp>
      <bmp x="442" y="319">../ressource/sprite/effect/ddd/36.bmp</bmp>
</action>
```

2. Présentation de Qt



1. Présentation

Qt a été développé par la société Trolltech, basée à Oslo en Norvège. Son développement a commencé en 1991 et il a été dès le début utilisé par KDE, un des principaux environnements de bureau de Linux.

Qt signifie "Cute", ce qui signifie "Mignonne", parce que les développeurs trouvaient que la lettre Q était jolie dans leur éditeur de texte.

Il s'agit d'une bibliothèque multi-plateforme pour créer des GUI (Graphic User Interface – programme sous forme de fenêtre), écrite en C++ et faite pour être utilisée à la base en C++. Cependant, il est aujourd'hui possible de l'utiliser dans d'autres langages comme Java, Python, etc.

Qt est constituée d'un ensemble de bibliothèques, appelées "modules". On peut y trouver entre autres ces fonctionnalités :

- **Module GUI** : c'est toute la partie création de fenêtres. C'est ce que nous avons utilisé durant ce projet
- **Module OpenGL** : Qt peut ouvrir une fenêtre contenant de la 3D gérée par OpenGL.
- **Module de dessin** : Module permettant de dessiner dans une fenêtre 2D.
- **Module XML** : Permet d'échanger des données avec des fichiers formés à l'aide de balises, un peu comme le XHTML.
- **Module SQL** : permet un accès aux bases de données (MySQL, Oracle, PostgreSQL...).

2. Pourquoi Qt ?

Dans le cadre de notre formation, nous avons décidé de n'utiliser que des logiciels libres, c'est pourquoi nous nous sommes tournés vers Qt, une bibliothèque de création de GUI.

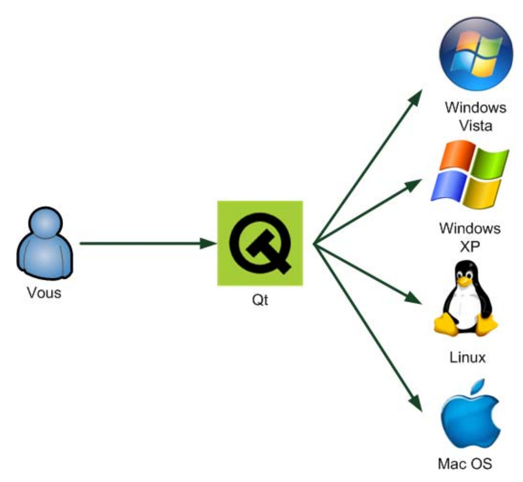
Cependant, il existe également une licence payante de Qt.

En effet, à l'origine, Qt possédait une licence propriétaire, son code source était fermé. Cependant, aujourd'hui, Trolltech propose une double licence :

- Si on fait un programme libre (ce qui est notre cas), alors on peut utiliser la version libre de Qt gratuitement.

- Si on fait un programme propriétaire, alors on doit acheter une licence auprès de Trolltech.

Nous avions également décidé de faire de notre programme, une application multi-plateforme, et pour cela, Qt semblait tout désigné.



De ce fait, Qt est également capable de créer une fenêtre qui adopte de look de l'OS (Operating System), sur lequel on se trouve. Ainsi, sur Linux, la fenêtre disposera de la même apparence que les fenêtres de votre OS, alors que sur Windows, le programme agira de la même manière en adoptant l'apparence d'une fenêtre Windows.

Qt est également un logiciel dont on ne parle pas beaucoup, mais qui n'en est pas moins utilisé par les grandes firmes de l'informatique. Ainsi, on peut retrouver du Qt dans des logiciels appartenant à Archos, Adobe ou encore Skype.

Dernier point intéressant, Qt dispose suite à son installation de beaucoup d'éléments intéressants pour le développement d'un GUI. Ainsi, on peut retrouver une foule d'exemple de programmes, expliqués de manière détaillée, afin de bien comprendre comment Qt fonctionne. On dispose également d'un assistant, un grand manuel d'utilisation informatisé, regroupant toutes les informations nécessaires au développement d'une application avec Qt. Et enfin, il dispose également d'un logiciel de design, appelé Qt Designer, qui va beaucoup servir, permettant de créer de manière graphique sa fenêtre, avant d'attaquer le codage.

3. Les Avantages de Qt

Il permet de :

- créer efficacement un GUI pour un programme.
- développer une application multi-plateforme
- développer une application avec du matériel open source.
- disposer d'outils efficaces, comme des exemples détaillés, un assistant, et une application de design.

4. Les Inconvénients de Qt

- Qt étant une librairie du C++, il est indispensable de maîtriser un minimum ce langage de programmation.
- Qt disposant de nombreux modules, il nécessite un certain temps de documentation, afin de trier ce qui est utile ou non et de se familiariser avec les différentes fonctions contenues dans sa librairie.
- Qt permet d'utiliser le html avec certains de ses widgets, comme par exemple le 'textbrowser', ce qui pouvait être un très bon point pour une mise en page propre couplée à une feuille de style en .css. Or, toutes les fonctionnalités du html ne sont pas prises en compte par Qt et il est impossible d'associer une feuille de style css aux données html, ce qui fait que nous n'avons pas les possibilités graphiques, que nous aurions pu espérer.

5. Qt Designer

Qt Designer, est donc le logiciel de Design fournit avec Qt, permettant de créer de façon graphique, comme par exemple Visual Studio, nos fenêtres.

3. Utilisation du mode console en C++ pour le serveur {Olivier LONZI}

1. Pourquoi du C++ en mode console

Le mode console permet de simplifier l'inter-opérabilité entre Linux et Windows, comme nous l'avions choisis dans notre cahier des charges, sans avoir recours à des sur-couche lourde comme QT, de plus, aucune action n'est nécessaire pour l'administrateur du serveur (voir Manuels d'utilisation du serveur de jeu p 52). Une administration via des pages web (plus intuitive et possible à distance) est développée par Jean-François BERNARD, voir sa partie p 51.

Pourquoi l'utilisation du C++ pour un serveur ?

Le C++ présente certes des temps d'exécution plus long qu'un programme compilé en C et est donc peut recommandé pour la création d'un serveur. Cependant pour la modélisation et pour sa simplicité de programmation, et surtout comme le serveur ne nécessitait pas d'avoir un temps de réponse très plus court, j'ai choisis le C++ comme langage.

Pourquoi pas un autre langage que C, ou C++ ?

Pour ne pas avoir à créer un API pour le client.

2. Les avantages du C++

Le C++ présente un certain nombre d'avantages pour moi :

- Gratuité du compilateur et liberté d'utilisation des exécutables après compilation.
- Je connais le langage depuis 2 ans.
- Possibilité de trouver toutes sortes de librairies à intégrer, comme par exemple 'mysql++', pour gérer les BdD MySql.
- Dans notre cas, QT et SDL peuvent être utilisés avec le C++, donc nous évite de recourir à des APIs pour l'intégration, tout en utilisant un système proche.

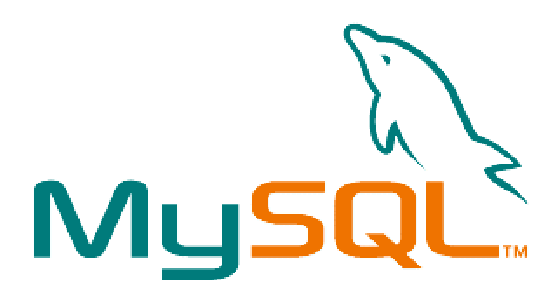
3. Les inconvénients du C++

Le C++ présente un certain nombre d'inconvénients dans son ensemble :

- Plus lent en exécution que le C.
- Langage permissif, facilité d'avoir des fuites mémoire non détectées.

Malgré ces inconvénients j'ai retenu ce langage, comme expliqué dans le chapitre «Pourquoi du C++ en mode console».

4. Utilisation de MySQL {Olivier LONZI & Jean-François BERNARD}



1. Présentation

MySQL est un système de gestion de base de données relationnelles SQL de la société MySQL AB (rachetée en janvier 2008 par Sun Microsystems, propriétaire de Oracle) conçu sur le système d'une licence d'utilisation payante si cette dernière est à un but lucratif.

Il existe de nombreuses librairies permettant l'utilisation de MySQL dans la majeure partie des langage de programmation (ex: PHP, C++, ASP) ainsi que des systèmes d'exploitation (ex: Windows, Unix, FreeBSD). Ce système est très populaire dans le milieu du web grâce à son utilisation pour les plateformes LAMP (Linux-Apache-MySQL-PHP).

2. Pourquoi MySQL

Nous avons choisi le rangement de nos données dans des bases de type MySQL, pour les raisons suivantes :

- Gratuité du SGBD pour une utilisation à but non lucratif.
- Possibilité de lier MySQL avec le serveur sur simple installation de la libraire «mysql++»
- Actuellement l'une des bases de données les plus utilisées avec le moteur de page web «PHP»
- Connaissance dans son administration en ligne de commande.
- Connaissance dans l'utilisation de requêtes SQL.

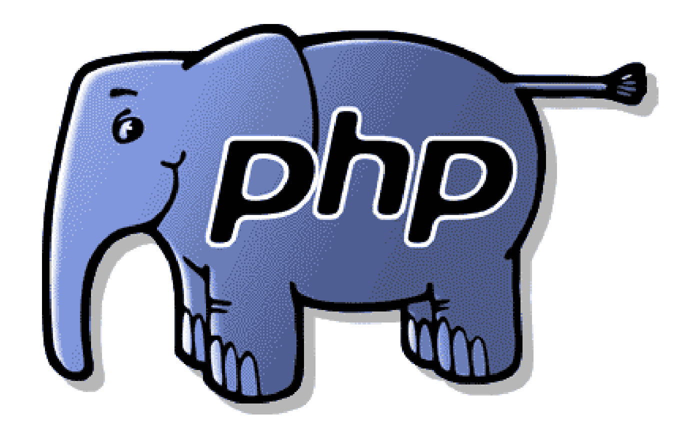
3. Les avantages de MySQL

- Gratuité des composants pour une utilisation à but non lucratif.
- Possibilité de lier MySQL avec le serveur sur simple installation de la libraire «mysql++»
- Peut être lié à distance, en utilisant une adresse IP et un port.

4. Les inconvénients de MySQL

- Un des SGBD les plus lents, pour des tables avec énormément de données à l'instar du très connu ORACLE.

5. Utilisation de PHP {Jean-François BERNARD}



1. Présentation

PHP (Hypertext Preprocessor) est un langage de script créé en 1994 par Rasmus Lerdorf sous la forme d'une bibliothèque logicielle en Perl, puis sous la forme d'une implémentation du langage C. Le cœur de cette version a part la suite été remanié par deux étudiants en 1997 afin de lui donner la forme qu'on lui connaît actuellement sachant que le modèle objet n'est disponible que depuis sa version 5.

Ce langage est très populaire dans le milieu du web car il permet la création de page Internet dynamique. De plus, son utilisation dans les plateformes LAMP lui a permis de se démocratiser dans la majeure partie des site Internet.

2. Pourquoi PHP

Nous avons choisit la création du site web à l'aide de cette technologie pour les raisons suivantes:

- Gratuité de l'utilisation du langage PHP
- Actuellement l'un des moteurs de page web les plus utilisé avec les base de données MysSQL
- Connaissance dans son utilisation pour la création de site Internet

3. Les avantages de PHP

- Programmation proche du langage C
- Possibilité d'utiliser le modèle objet
- Facilité d'accès aux bases de données MySQL
- Nombreuses fonctions préexistantes

4. Les inconvénients de PHP

- Présence de quelques failles de sécurité
- Présence de différences dans les fonctions PHP selon la version utilisée

6. Les manuels

1. Serveur de jeu

2. Manuel d'installation

1. Besoin matériel

- Afin d'installer le serveur, il faut disposer d'une machine connectée en permanence à internet, branchée derrière un simple modem, ou un routeur permettant le routage NAT des ports.
- La machine utilisée doit tourner sur un système d'exploitation 32 bits, de type Windows NT ou Linux.

2. Choix du type d'installation

- Il existe plusieurs types d'installation
  - Le serveur installé sur la machine utilise une base de données distante.
  - Le serveur installé sur la machine utilise une base de données locale.

3. Installation pas à pas (Windows NT – 32bits)

- **Attention :** Si vous utilisez une base de données distante, veuillez passer au grand point suivant, sans détailler les sous-parties de ce point.
  - Copier le dossier «\ServeurMySQL» fournit avec le CD et/ou disponible sur le site du jeu sur votre disque dur (ex : «C:\ServeurMySQL»)
  - Modifier le fichier «C:\ServeurMySQL\my.ini», puis modifier les informations suivantes :
    - (ligne 53~) port=3306 (Choisissez votre numéro de port, 3306 par défaut)
    - (ligne 68~) port=3306 (Même numéro que choisit ci-dessus)
    - (ligne 72~) basedir="C:\ServeurMySQL"
    - (ligne 75~) datadir="C:\ServeurMySQL/mysql/Data/"
  - Installer le serveur sous forme de service (Windows NT)
    - Placer vous dans le répertoire copié «C:\ServeurMySQL»
    - Puis dans le répertoire «\bin»
    - Taper la commande :
      «mysqld-nt.exe --install MySQL5_IrisFight --defaults-file="C:\ServeurMySQL\my.ini"»
    - Et enfin lancer le serveur à l'aide de la commande : «net start MySQL5_IrisFight»
      - Si une erreur survient vis à vis du port, re-modifier le fichier «my.ini» pour changer le port utilisé.
  - **Info :** Vous disposerez d'une base de données vide, avec un compte administrateur par défaut (login : **admin**, mot de passe : **irisfight**), pensez à le changer.
- Une fois la base de données disponible, il faut configurer et installer le serveur de jeu.
  - Copier le dossier «\ServeurIrisFight-win32» depuis le CD ou le site web, sur votre disque dur «ex : C:\ServeurIrisFight».
  - Modifier le fichier «C:\ServeurIrisFight\config.ini»

      - dans la section [BdD]
        - ip=localhost (ou dns du serveur distant)
        - port=3306 (ou tout autre port suivant celui choisit précédemment)
        - base=iris-fight (nom par défaut si vous avez installé la BdD avec les étapes précédentes)
        - user=root
        - pass=
      - dans la section [SERVEUR]
        - port=1664 (ou tout autre numéro de port)
        - max_client=100 (indique le nombre maximum de client connectés simultanément)
    - Lancer le serveur.
      - Taper la commande «IFServeur.exe» pour lancer le serveur.
        - **Attention :** Les numéros de port doivent être compris entre 1 et 65535, de plus il est déconseillé d'utiliser les port inférieurs à 1000, car il sont standardisés.
        - **Info :** Vous ne pourrez pas lancer plusieurs instances simultanément sur le serveur, mais vous pouvez dupliquer le répertoire «C:\ServeurIrisFight», pour instancier un nouveau serveur, avec une autre bdd.
- Maintenant que le serveur est lancé, il faut le tester.
    - Pour ce faire, lancer le client local en utilisant votre compte administrateur. (se référer au chapitre d'utilisation du client)
    - Si vous souhaitez administrer la BdD, il faudra utiliser l'application web (se référer au chapitre d'utilisation de l'application web)
- Une fois les tests effectués et concluant en local, nous allons permettre l'accès à la machine depuis l'extérieur.
    - 1er cas : Vous êtes connecté à internet directement avec un modem RTC ou un routeur branché en USB.
      - Aucune manipulation sur les ports n'est nécessaire à ce niveau.
    - 2ème cas : Vous utilisez un routeur gérant les redirection NAT.
      - Référer vous au manuel de configuration NAT de votre routeur pour rediriger les ports utiles :
        - 3306 – par défaut – pour la bdd, si vous l'avez installé en local
        - 1664 – par défaut – pour le serveur de jeu.
    - Dans tout les cas :
      - Vérifier qu'aucun firewall ne bloque les port utiles :
        - 3306 – par défaut – pour la bdd, si vous l'avez installé en local
        - 1664 – par défaut – pour le serveur de jeu
        - Vous pouvez divulguer votre adresse ip internet (ou votre dns) aux personnes susceptibles de jouer sur votre serveur.

4. Installation pas à pas (Linux – 32bits)

- **Attention :** Si vous utilisez une base de données distante, veuillez passer au grand point suivant, sans détailler les sous-parties de ce point.
  - Suivant votre distribution les commandes d'installation devraient permettre d'installer la dernière version de MySQL :
    - Debian : apt-get install mysql-server
    - Gentoo : emerge mysql-server
    - En cas de problème, référez vous au site officiel de MySQL : «http://dev.mysql.com/doc/refman/5.0/fr/installing-binary.html»
  - Une fois MySQL installée, taper la commande MySQL , pour vous connecter en ligne de commande au serveur MySQL.
    - Taper la commande «CREATE DATABASE 'iris-fight';» pour créer la base de données de jeu.
      - **Attention :** N'oubliez pas les points-virgules finaux.
    - Puis «USE 'iris-fight';»
      - «CREATE TABLE `persos` (
        `ID_perso` int(10) unsigned NOT NULL auto_increment,
        `actif` int(1) NOT NULL default '0',
        `admin` int(1) NOT NULL default '0',
        `joueur_prenom` varchar(50) NOT NULL default '',
        `joueur_nom` varchar(50) NOT NULL default '',
        `nom` varchar(50) NOT NULL default '',
        `pass` varchar(50) NOT NULL default '',
        `courriel` varchar(100) NOT NULL default '',
        `niveau` int(2) unsigned NOT NULL default '1',
        `experience` int(10) unsigned NOT NULL default '0',
        `point_vie` int(2) unsigned NOT NULL default '1',
        `point_orbe_1` int(2) unsigned NOT NULL default '0',
        `point_orbe_2` int(2) unsigned NOT NULL default '0',
        `point_orbe_3` int(2) unsigned NOT NULL default '0',
        `point_orbe_4` int(2) unsigned NOT NULL default '0',
        `point_orbe_5` int(2) unsigned NOT NULL default '0',
        `point_orbe_6` int(2) unsigned NOT NULL default '0',
        PRIMARY KEY (`ID_perso`),
        UNIQUE KEY `nom` (`nom`)
) TYPE=InnoDB;».
    - Nous allons ici ajouter un compte administrateur de jeu avec la commande : «INSERT INTO 'persos' ('nom', 'pass', 'admin') VALUES('**admin**', '**irisfight**', 1);»
- Une fois la base de données disponible, il faut configurer et installer le serveur de jeu.
  - Copier le dossier «/ServeurIrisFight-linux32» depuis le CD ou le site web, sur votre disque dur «ex : /home/*vous*/ServeurIrisFight».
  - Modifier le fichier «/home/*vous*/ServeurIrisFight/config.ini»
    - dans la section [BdD]
      - ip=localhost (ou dns du serveur distant)

        - port=3306 (ou tout autre numéro de port suivant celui choisit précédemment)
        - base=iris-fight (nom par défaut si vous avez installé la BdD avec les étapes précédentes)
        - user=root
        - pass=
      - dans la section [SERVEUR]
        - port=1664 (ou tout autres numéro de port)
        - max_client=100 (indique le nombre maximum de client connectés simultanément)
    - Lancer le serveur.
      - Taper la commande «/home/*vous*/ServeurIrisFight/IFServeur» pour lancer le serveur.
        - **Attention :** Les numéros de port doivent être compris entre 1 et 65535, de plus il est déconseillé d'utiliser les port inférieurs à 1000, car ils sont standardisés.
        - **Info :** Vous ne pourrez lancer plusieurs instances simultanément sur le serveur, mais vous pouvez dupliquer le répertoire «/home/*vous*/ServeurIrisFight», pour instancier un nouveau serveur, avec une autre bdd.
- Maintenant que le serveur est lancé, il faut le tester.
    - Pour ce fait, lancer le client local en utilisant votre compte administrateur. (se référer au chapitre d'utilisation du client)
    1. Si vous souhaitez administrer la BdD, il faudra utiliser l'application web (se référer au chapitre d'utilisation de l'application web)
- Une fois les tests effectués et concluant en local, nous allons permettre l'accès à la machine depuis l'extérieur.
    - 1er cas : Vous êtes connecté à internet directement avec un modem RTC ou un routeur branché en USB.
      - Aucune manipulation sur les ports n'est nécessaire à ce niveau.
    - 2ème cas : Vous utilisé un routeur gérant les redirection NAT.
      - Référer vous au manuel de configuration NAT de votre routeur, pour rediriger les ports utiles :
        - 3306 – par défaut – pour la bdd, si vous l'avez installé en local
        - 1664 – par défaut – pour le serveur de jeu.
    - Dans tout les cas :
      - Vérifier qu'aucun firewall ne bloque les port utiles :
        - 3306 – par défaut – pour la bdd, si vous l'avez installé en local
        - 1664 – par défaut – pour le serveur de jeu.
- Vous pouvez maintenant divulguer votre adresse IP internet (ou votre dns) aux personnes susceptibles de jouer sur votre serveur.

3. Manuel d'utilisation

Aucune utilisation particulière n'est à effectuer sur le serveur de jeu, une fois celui-ci installé.

4. Site Internet

5. Manuel d'installation

1. Besoin matériel

- Afin d'installer le site Internet, il faut avoir préalablement installé sur une machine les logiciels Apache, PHP et MySQL.

2. Installation pas à pas

- Copier le dossier «\SiteInternet» fournit avec le CD et/ou disponible sur le site du jeu sur votre disque dur (ex : «C:\SiteInternet»)
- Modifier le fichier «C:\SiteInternet\script\conf.php» afin de configurer
  - (ligne 9~) define('DB_HOST', 'localhost'); (Si l'utilisation est autre que locale, mettez à la place de «localhost» l'ip hébergeant et le port de la base de donnée utilisé pour le jeu)
  - (ligne 10~) define('DB_NAME', 'iris-fight'); (A la place d'«iris-fight», mettez le nom de la base de donnée utilisée)
  - (ligne 12~) define('DB_USER', 'iris-fight'); (A la place d'«iris-fight», mettez le nom de l'utilisateur ayant les droits de sélection, insertion et mise à jour pour la base de donnée définie ci dessus)
  - (ligne 11~) define('DB_PASS', 'iris'); (A la place d'«iris», mettez le mot de passe de l'utilisateur que vous avez définie ci dessus)

6. Manuel d'utilisation

1. Créer un compte utilisateur

Pour créer un compte, vous devez aller dans la partie «Compte» du site Internet et ensuite cliquer sur le lien «Créer un compte». Dedans, vous devez rentrer vos noms, prénoms, email (utilisé pour l'envoi du mot de passe), nom du personnage et mot de passe (de 6 caractères minimum).

2. Gestion du compte

Pour gérer votre compte, vous devez vous être connecté à ce dernier. Pour cela, il vous suffit d'aller dans la partie «Compte» et d'y entrer votre nom de personnage et votre mot de passe, puis de valider. Dedans, vous avez la possibilité de modifier vos informations personnelles en modifiant les champs prévus à cet effet. De plus, vous pouvez aussi distribuer les points d'orbe qui sont disponibles à l'aide des boutons «+» et «-» situés de part et d'autre de vos points d'orbe actuels .

3. Accéder à l'interface d'administration

Pour accéder à l'interface administrateur, il suffit de se connecter au site Internet à l'aide d'un compte administrateur et, en bas de cette page, de cliquer sur le lien vers cette interface.

4. Désactiver/Activer un compte

Vous avez la possibilité de choisir l'activation ou non d'un utilisateur en cliquant sur la partie «Activation/Désactivation de compte» de l'interface d'administration. Dedans, une liste des comptes est

affichée et il ne vous reste plus qu'à cliquer sur le bouton «Activer/Désactiver» de l'utilisateur souhaité.

5. Passer un utilisateur au statut d'administrateur

Vous avez la possibilité de modifier les privilèges d'un utilisateur en cliquant sur la partie «Comptes administrateur» de l'interface d'administration. Dedans, une liste des administrateurs est affichée et il ne vous reste plus qu'à cliquer sur le bouton «Enlever» de l'administrateur souhaité pour le transformer en utilisateur. A l'inverse, il suffit de rentrer le nom d'un utilisateur dans le champ en bas de page et de cliquez sur «OK» afin de voir l'utilisateur sélectionné devenir administrateur.

7. Client

8. Manuel d'installation

1. Windows (32 bits)

- Copier le dossier «\ClientWin32» fournit avec le CD et/ou disponible sur le site du jeu sur votre disque dur (ex : «C:\iris-fight\»)
- Exécuter «C:\iris-fight\iris-fight.exe»

2. Linux (32 bits)

- Copier le dossier «\ClientLinux32» fournit avec le CD et/ou disponible sur le site du jeu sur votre disque dur (ex : «/home/vous/iris-fight/»)
- Se placer dans le répertoire : « cd /home/vous/iris-fight/»
- Exécuter la commande : «make»
- Exécuter la commande : «make install»
- Taper la commande : «iris-fight» pour lancer le jeu.

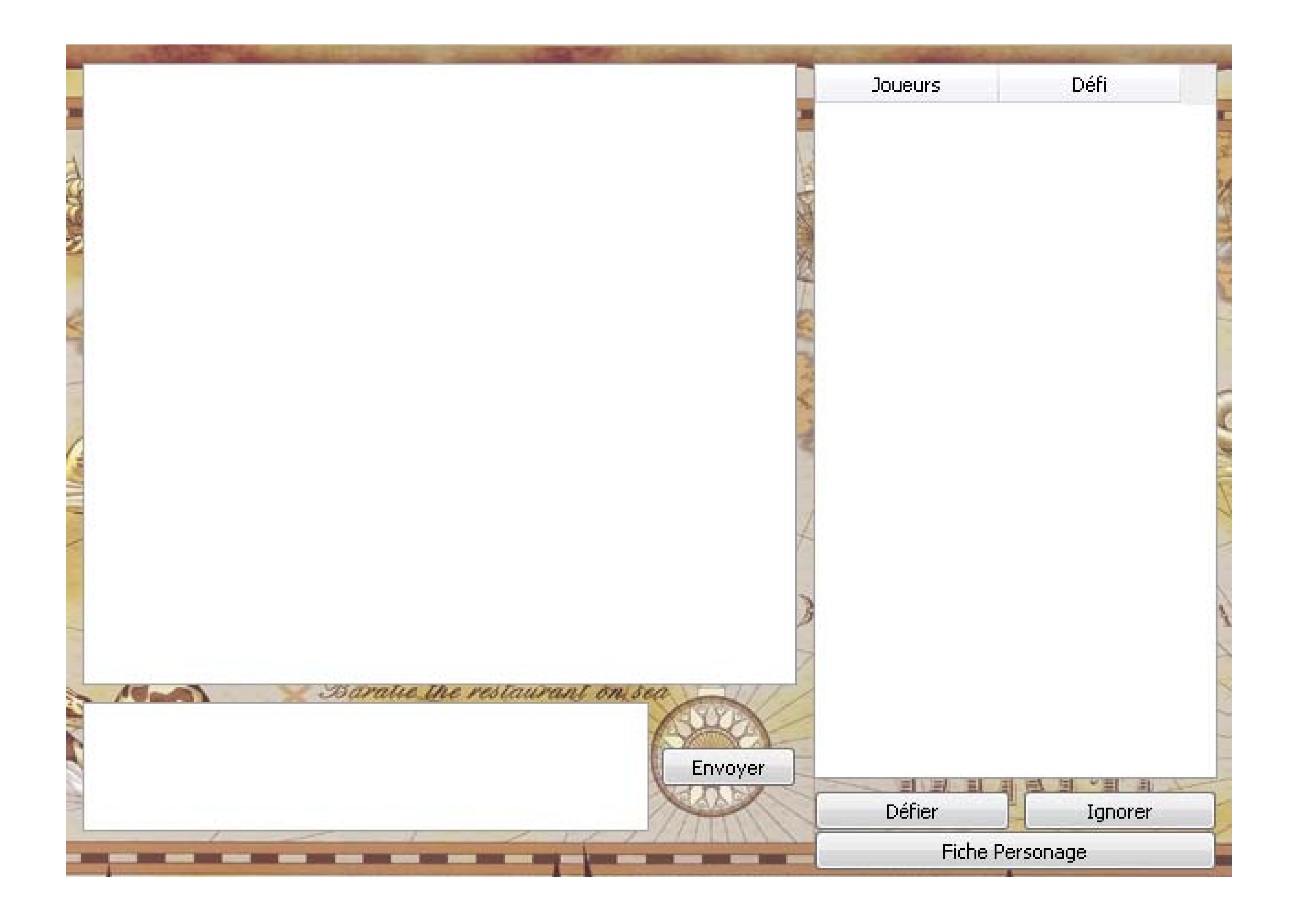
9. *Manuel d'utilisation*

1. Connexion



Le client entre son login, son mot de passe, et l'adresse du serveur auquel il souhaite se connecter. Puis il valide son choix en cliquant sur le bouton Connect.

2. Fenêtre de discussion



Après s'être connecté et identifié à un serveur de jeu, le client arrive sur la page de chat, lui permettant de dialoguer avec les utilisateurs dans un salon de discussion général se situant sur la gauche.

Dans la partie supérieure gauche se trouve le champs de discussion, où apparaissent les messages des joueurs qui souhaitent dialoguer. Dans la partie inférieur gauche, se situe le champs de saisie du message, qui permet au client d'envoyer son message aux autres joueurs dans le salon de discussion.

Sur la partie droite de la fenêtre, l'utilisateur peut voir la liste des clients connectés et décider de proposer/refuser un duel avec ceux-ci, en cliquant sur les boutons correspondants.

Enfin, dans la partie inférieure droite, l'utilisateur peut accéder à sa fiche de compétence, en cliquant sur le bouton correspondant.

Si un défi est accepté, l'utilisateur accédera directement dans la partie Duel du jeu.

3. Fiche de personnage



Dans cette fenêtre, l'utilisateur peut voir le nom de son personnage dans la partie supérieure de la fenêtre, suivi de son niveau et de la liste de ses compétences.

Dans la partie inférieure de la fenêtre, l'utilisateur peut cliquer sur le bouton « Chat Room » pour accéder au Chat général et faire des duels.

En cas de passage de niveau, l'utilisateur peut modifier ses compétences, en attribuant les points se situant dans la partie « Points Disponibles », à la compétence qu'il souhaite améliorer. Pour augmenter les points de compétence, il suffit de cliquer sur le bouton « + », et pour diminuer les points compétence, il suffit d'appuyer sur le bouton « - ».

4. Défi



Le joueur 1 décide des orbes à sélectionner pour sa première action et valide sa première action en sélectionnant l'option "valider". Il fait ensuite ses autres actions de la même manière et valide la fin de son tour en cliquant sur le bouton "Fin de tour".

Le joueur 2 fait exactement la même manipulation au même moment, et une fois les deux joueurs prêts, le serveur affiche le résultat du tour.

Le combat se termine lorsque la vie d'un joueur atteint zéro, ou que l'un des deux joueurs quitte la partie prématurément.

7. Cahier des charges fourni à l'IUT le 20 décembre 2007

1. Présentation du projet

1. But

Ce projet consiste à réaliser un jeu en deux dimensions dans lequel deux joueurs s'affrontent. Ce jeu sera multi-plateformes, c'est à dire qu'il devra fonctionner sur les plateformes traditionnelles (Windows, Linux etc...).

Les joueurs se connectent à un serveur de combat, dans lequel ils créent ou ont déjà créé un personnage.

Une fois que les joueurs ont choisi le personnage qui les représentera, ils apparaissent dans la liste des joueurs connectés et peuvent proposer un combat à tous les autres joueurs de la liste.

Une fois qu'un joueur en a défié un autre et que celui-ci accepte le duel, un combat commence entre leurs deux personnages.

L'issu du duel se détermine par la perte de tout les points de vie d'un des deux personnages, ou par l'abandon d'un des joueurs.

Une fois le duel terminé, des points d'expériences sont gagnés, permettant ainsi aux joueurs de faire évoluer leurs personnages.

2. Détail rapide du système de jeu

Lors des combats les joueurs récupèrent des orbes au début des tours, puis ils peuvent les consommer afin de réaliser des sorts. Un sort de protection coûte moins d'orbes qu'un sort offensif. Les tours se déroulent en trois phases :

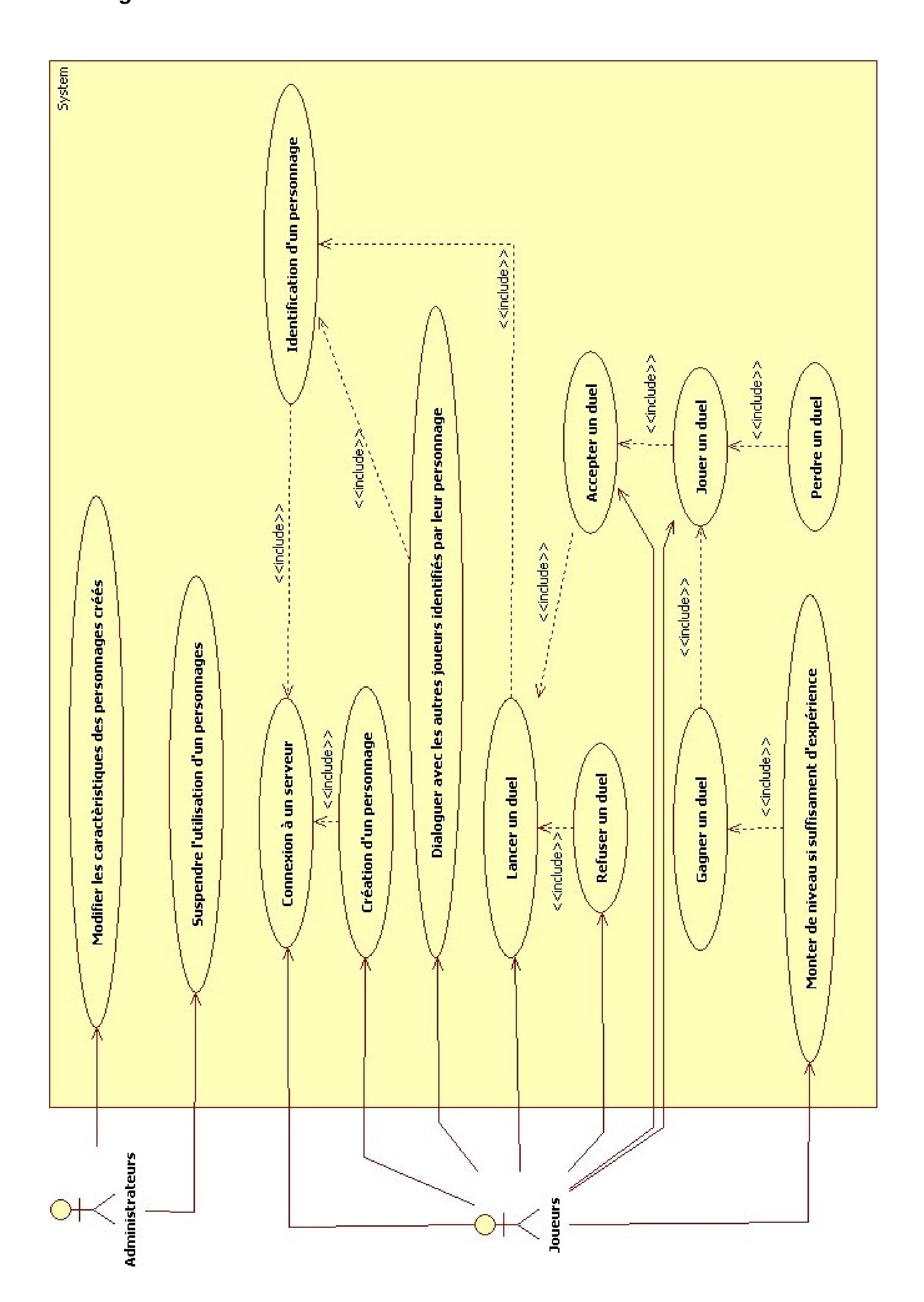
- Phase Préparation : Les joueurs consomment leurs orbes (ou non).
- Phase Défensive : Les sorts défensifs sont lancés.
- Phase Offensive : Les sorts offensifs sont lancés.

Une fois les trois phases d'un tour terminées, un nouveau tour commence, sauf si les conditions de fin de partie sont atteintes.

3. Choix du langage de programmation

Nous avons choisi pour réaliser ce jeu de programmer exclusivement en C++, avec la librairie SDL pour l'interface de jeu, et la librairie Qt pour les autres interfaces graphiques.

4. Diagramme des cas d'utilisation



2. Études ou réalisations préalables

Avant de débuter notre projet, nous avons fait toutes sortes de recherches afin d'être sûr qu'il serait faisable, abordable et homogène. Pour cela, nous avons porté notre attention sur ces différents points :

- Librairies de programmation : Pour faire ce projet, nous avons opté pour certains langages utilisant des librairies que nous n'avons pas (ou peu) vu, durant nos formations respectives. De ce fait, nous avons fait des recherches pour comprendre ces librairies (Comme la SDL), et nous assurer que nous serions en mesure de mener à bien notre projet.
- Aspect graphique du projet : Afin de disposer d'une bonne ergonomie et d'un graphique unique, nous nous sommes mis d'accord sur l'aspect graphique que nous voulions donner à notre projet, de manière à ne pas avoir à refaire une étape qui s'avère longue et non primordiale.

3. Liste des livrables attendus

À la fin de notre projet, nous serons en mesure de fournir les éléments suivants :

- Exécutable du serveur de jeu (Windows et Linux).
- Exécutable du client (Windows et Linux).
- Documents permettant la mise en place du serveur.
- Documents expliquant comment lancer le client et comment jouer.
- Rapport de projet comprenant toutes notre démarche de création du système.

4. Les participants du projet et leurs objectifs

Jean-Yves MARIN :

- Réalisation de l'arène de combat (C++ avec librairie SDL)
  - Récupération des actions du client.
  - Graphisme du jeu et effet sonore.
  - Invention des règles du jeu.

Georges-Édouard BOUCHER :

- Réalisation des interfaces graphique type Window (C++ avec Qt).
  - Demande des identifiants pour la connexion au serveur.
  - Création des personnages.
  - Fenêtre de discutions, avec liste des joueurs et possibilité de lancer des défis.
  - Interface graphique de l'écran d'accueil et du chat.

Olivier LONZI :

- Réalisation du serveur de jeu (C++, en mode console)
  - Création du service de communication avec les clients (Socket TCP/IP).
  - Gestion des règles de jeu.

5. Tableau des risques

Risques potentiels	Effets sur les objectifs	Phase d'identification	Mesures préventives	Mesures curatives	Niveau de risque
Retard dans un module particulier.	Retard du sur la sortie du projet.	Dépassement du planning.	Respecter le planning.	Travailler de façons plus organisée et en avance sur le planning maximum.	Élevé
Perte de données.	Travail à refaire.	Données absentes.	Sauvegardes régulières.	Appliquer la sauvegarde.	Faible
Application des connaissances.	Retard sur le planning.	Trop de temps pour faire quelque chose.	Consolider les bases.	Se mettre à jour sur les bases.	Moyen
Bugs.	Produit instable.	Bugs trop fréquents.	Vérifications régulières.	Débugage.	Moyen
Matériel déficient.	Retard sur le planning.	Matériel inutilisable.	Vérifier le matériel bien avant la date limite.	Réinstaller/Réparer.	Elevé